UNIVERSITY OF BREMEN
FACULTY 7: BUSINESS STUDIES & ECONOMICS
CHAIR OF LOGISTICS

PHD THESIS

Container Hinterland Drayage

On the Simultaneous Transportation of Containers Having Different Sizes

Author:
Dipl.-Math. Julia Funke
Bremen

Submitted to:
Promotionsausschuss Dr. rer. pol.
University of Bremen

Reviewers:
Prof. Dr.-Ing. Herbert Kopfer
Prof. Dr. Tobias Buer

Date of Colloquium:
October 23, 2017

Acknowledgment

First of all, I would like to thank my supervisor Prof. Herbert Kopfer for the chance to work on this interesting topic at the Chair of Logistics at the University of Bremen as well as my entire dissertation committee; I am especially grateful to Prof. Tobias Buer for being the second reviewer of my thesis.

I want to thank my family and my dear friends for the constant support during the process. And I want to particularly thank my colleagues as well as my former employers, colleagues and friends who gave me the chance to learn about the topic: INFORM GmbH, who gave me a deep insight into logistic decision-making in the field of route planning, and the Research Institute for Discrete Mathematics of the Rheinische Friedrich-Wilhelms Universität Bonn, where I had the chance to improve my programming skills and mathematical expertise. Apart from that, I had some great times and many interesting discussions on several conferences that encouraged me to do my thesis and took my ideas forward.

I hope you enjoy your reading.

Julia Funke

Contents

Contents v

List of Figures ix

List of Tables xi

Abbreviations xiii

1 Introduction 1

1.1 Containerization . . . 2

1.1.1 History . . . 2

1.1.2 Maritime Container Transportation . . . 3

1.2 Intermodal Freight Transportation . . . 5

1.2.1 Container Hinterland Drayage . . . 6

1.2.2 Actors . . . 7

1.2.3 Empty Container Repositioning . . . 9

1.3 Objectives and Structure of the Thesis . . . 10

2 Basic Definitions 13

2.1 Combinatorial Optimization Problems . . . 13

2.1.1 Linear, Integer and Mixed Integer Programming . . . 14

2.1.2 Graphs . . . 14

2.1.3 Flow Problems . . . 17

2.1.4 Routing and Scheduling Problems . . . 21

2.1.5 More Mathematics . . . 24

2.2 Algorithms . . . 25

2.2.1 Complete Algorithms/Exact Algorithms . . . 25

2.2.2 Approximate Algorithms/Heuristics . . . 25

2.2.3 Metaheuristics . . . 26

3 Literature Overview **29**
3.1 Classification of Drayage Problems . . . 32
3.1.1 (Multiple) Traveling Salesman Problem . . . 32
3.1.2 Vehicle Routing Problems with Synchronization . . . 33
3.2 Full Truckload Problem . . . 34
3.3 Repositioning of Empty Containers . . . 37
3.3.1 No Separation . . . 39
3.3.2 Separation . . . 44
3.4 Multi-size Container Transportation . . . 50
3.5 Challenges . . . 56

4 Multi-size Container Inland Transportation **59**
4.1 Problem Definition . . . 59
4.1.1 Hinterland Requests . . . 60
4.1.2 Instance . . . 62
4.2 Solution . . . 63

5 Exact Approach **65**
5.1 Graph Definition . . . 66
5.1.1 Hinterland Requests . . . 67
5.1.2 (Un-)loading Operations . . . 68
5.1.3 Container Storage Operations . . . 71
5.1.4 Entire Graph Representation . . . 72
5.2 Assigning Containers . . . 72
5.2.1 Decision Variables . . . 74
5.2.2 Balance Values . . . 74
5.2.3 Capacities . . . 75
5.2.4 Entire Model . . . 75
5.3 Building Routes . . . 76
5.3.1 Decision Variables . . . 77
5.3.2 Entire Model . . . 78
5.4 Coupling of the Models . . . 78
5.5 Objectives . . . 80

6 Heuristic Approach **81**
6.1 Models for Containers . . . 82
6.1.1 Connected Components . . . 82
6.1.2 Graph . . . 84
6.1.3 Assignment of Empty Containers . . . 85
6.1.4 Assignment of Empty and Fully Loaded Containers . . . 88
6.2 Heuristics for Truck Routes . . . 90

6.2.1 Preliminary Considerations . . . 90
6.2.2 Construction of Routes . . . 93
6.2.3 Feasibility Check . . . 94
6.2.4 Estimations of Time Windows and Durations . . . 96
6.2.5 Evaluators . . . 98
6.3 Large Neighborhood Search . . . 98
6.3.1 Construction of the Initial Solution . . . 99
6.3.2 Improvement of the Initial Solution . . . 104
6.3.3 Simulated Annealing . . . 109
6.4 Implementation Details . . . 109
6.4.1 Partitioning of Large Instances . . . 109
6.4.2 Implementation of the Feasibility Check . . . 110
6.4.3 Route Assignment . . . 111
6.4.4 Computation Time of the Exact Approach . . . 111

7 Computational Study **113**
7.1 Test Instances . . . 113
7.1.1 Randomly Generated Instances . . . 114
7.1.2 Instances of Literature Sources . . . 115
7.2 Structure of the Analysis . . . 118
7.3 Exact Approach . . . 119
7.4 Heuristic Approach . . . 120
7.4.1 Parameter Adjustments . . . 121
7.4.2 Analysis of the Implemented Operators . . . 129
7.4.3 Solution Quality . . . 130
7.4.4 Application to Real-world Instances . . . 132
7.5 Summary of Findings . . . 136

8 Conclusion and Future Research **137**
8.1 Conclusion . . . 138
8.2 Future Research . . . 140

Appendix A Further Computational Results **143**

Bibliography **151**

Index **164**

List of Figures

1.1 Container Terminal Bremerhaven 2
1.2 Container Port Traffic, World 4
1.3 Container Port Traffic, EU and Germany 4
1.4 Typical Transportation Chain 5
1.5 Hinterland Drayage Operations 6

3.1 Full Truckload Problem 35
3.2 Street-Turn 38

4.1 Example Solution, mICTP 63

5.1 Graph Representation of Request Types 66
5.2 (De-)coupling Possibilities 68
5.3 Instance Graph 73
5.4 Balance Values and Capacities 77

6.1 Pickup and Delivery Tasks 83
6.2 Characteristic of Connected Components 87
6.3 Connected Components as Multi-Task Units 90
6.4 Precedence Constraints 91
6.5 Varying Duration of Connected Components 93
6.6 Distance Graph 96

7.1 Structure, Instance04 117
7.2 Development of Objective Values, 11 Hinterland Requests . . . 124
7.3 Development of Objective Values, 75 Hinterland Requests . . . 125
7.4 Development of Objective Values, 75 Hinterland Requests . . . 135

A.1 Development of Objective Values, 75 Hinterland Requests . . . 148
A.2 Development of Objective Values, 75 Hinterland Requests . . . 149

List of Tables

3.1 Overview of Literature Sources 56

4.1 Hinterland Requests 62

5.1 Request Nodes 70
5.2 Depot Nodes 71
5.3 Arcs 72

6.1 Balances, Pickup and Delivery Tasks 85

7.1 Instance Layout, 3-6 Hinterland Requests 114
7.2 Instance Layout, 11 Hinterland Requests 115
7.3 Instance Layout, 75 Hinterland Requests 117
7.4 Results Exact Approach, 3-6 Hinterland Request 119
7.5 Default Parameter Setting 121
7.6 Results Different Parameter Settings, 11 Hinterland Requests . . 122
7.7 Results Different Parameter Settings, 75 Hinterland Requests . . 127
7.8 Final Parameter Setting 128
7.9 Results Implemented Operators, 75 Hinterland Requests 129
7.10 Results, 11 Hinterland Requests 130
7.11 Detailed Results, DS2, DS7, DS8 131
7.12 Results Heuristic Approach, 75 Hinterland Requests 132
7.13 Detailed Results Heuristic Approach, Instance04, Instance07, Instance10, Instance13, Instance16, Instance21 134

A.1 Detailed Results Exact Approach, 3-6 Hinterland Requests . . . 144
A.2 Detailed Results Exact Approach, 3-6 Hinterland Requests . . . 145
A.3 Detailed Results, DS1, DS3, ... DS6, DS9 and DS10 146
A.4 Detailed Results Heuristic Approach, 75 Hinterland Requests . 147

Abbreviations

$\mathcal{NP}$	Non-deterministic Polynomial-time
$\mathcal{P}$	Polynomial-time
ALNS	Adaptive Large Neighborhood Search
amTSP	Asymmetric Multiple Traveling Salesman Problem
amTSPTW	Asymmetric Multiple Traveling Salesman Problem with Time Window Constraints
AP	Assignment Problem
APMCFP	Arc Partition Minimum Cost Flow Problem
APSP	All Pairs Shortest Path Problem
aTSP	Asymmetric Traveling Salesman Problem
BPP	Bin Packing Problem
CG	Column Generation
CO	Combinatorial Optimization
FIFO Technique	First In First Out Technique
FTL	Full Truckload
FTPDP	Full Truckload Pickup and Delivery Problem
FTPDPTW	Full Truckload Pickup and Delivery Problem with Time Window Constraints
GAP	Generalized Assignment Problem
ICT	Inland Container Transportation Problem
IE Request	Inbound Empty Hinterland Request

IF Request	Inbound Full Hinterland Request
IP	Integer Program
ISL	Institute of Shipping Economics and Logistics
LIFO Technique	Last In First Out Technique
LNS	Large Neighborhood Search
LR	Linear Relaxation
LTL	Less-than Truckload
MCFP	Minimum Cost Flow Problem
MCNDP	Minimum Cover of Nodes by Directed Paths
MFP	Multicommodity Flow Problem
mICTP	Multi-size Inland Container Transportation Problem
MIP	Mixed Integer Program
mTSPTW	Multiple Traveling Salesman Problem with Time Window Constraints
mTSPTWP	Multiple Traveling Salesman Problem with Time Windows and Precedences
OE Request	Outbound Empty Hinterland Request
OF Request	Outbound Full Hinterland Request
OR	Operations Research
OSC	Ocean Shipping Consultants
PDP	Pickup and Delivery Problem
PDPTW	Pickup and Delivery Problem with Time Window Constraints
RTS	Reactive Tabu Search
SA	Simulated Annealing
SCP	Set Covering Problem
SP	Shortest Path Problem
SPP	Set Partitioning Problem

TEU	Twenty-foot Equivalent Unit, standard container's size
TP	Transportation Problem
TS	Tabu Search
TSP	Traveling Salesman Problem
TSPTW	Traveling Salesman Problem with Time Window Constraints
VRP	Vehicle Routing Problem
VRPB	Vehicle Routing Problem with Backhauls
VRPMS	Vehicle Routing Problem with Multiple Synchronization Constraints
VRPTW	Vehicle Routing Problem with Time Window Constraints
WPB method	Window Partition Based Method

Chapter 1

Introduction

In an intermodal transportation chain *drayage* is the term used for the movement by truck of cargo that is filled in a loading unit. The most important intermodal transportation chain is the *intermodal container transportation*, in which containers represent the loading unit for cargo. Cost effectiveness constitutes a general problem of drayage operations: 25% to 40% of the total intermodal container transportation cost are accrued in drayage (Macharis and Bontekoning [2004]). The portion of inland costs are much higher, ranging from 40% to 80% of the total cost of (intercontinental) container shipping (Notteboom and Rodrigue [2005]). These costs stand in high contrast to the low portion of total distance crossed in seaport *hinterland* regions (inland area served by a certain port) of intercontinental container transportation chains, which is approximately 3% of the total distance traveled. The large range of hinterland connections constitutes a permanent challenge for fluent container traffic (Hildebrand [2008]). A major cost driver within container transportation chains is the movement and repositioning of empty containers; estimations for the share of such *unproductive movements* of all containers being transported vary from 21% at sea to about 40% on land (Konings [2005]). The present thesis investigates the potential to reduce drayage costs. Two solution methodologies are developed for operating a fleet of trucks that transports containers of different sizes, which addresses a recent gap in research in seaport hinterland regions.

The remainder of this chapter is structured as follows: Sections 1.1 and 1.2 give an overview of containerization and intermodal freight transportation. Afterwards, Section 1.3 defines the scope of the present research and explains the structure of the thesis.

1.1 Containerization

These days containers that are normed according to some standards (mostly ISO 668) represent the most important transportation techniques that are used in international freight transportation. Maritime container traffic increased by more than 40% from the year 2000 to the year 2010 (Nordsieck et al. [2016]). Whereas nowadays the *containerization* of transportation processes is a worldwide success story, in the past it was a difficult process due to shipowners.

1.1.1 History

Until the middle of the 19th century, truck drivers had to spend whole nights at ports waiting for their trucks to be (un-)loaded. At the end of 1930, the hauler Malcolm P. McLean first tried to avoid (un-)loading operations of cargo from one means of transportation to another by starting to load complete trucks onto ships with the aim of transporting these trucks as close as possible to their specified points of destination. The invention of standardized containers and

Figure 1.1: Container terminal Bremerhaven.
Pictures from http://www.bremerhaven.de/ (accessed on 1st March 2016).

trailers made it possible to initially transport trailers loaded with containers, and later to transport containers only by ship. However, none of the ship owners were convinced by Malcolm P. McLean's idea, which led to McLean giving up America's second largest trucking company and subsequently founding the company Sea-Land Inc. and becoming a shipowner himself. Within a short period of time Sea-Land Inc. successfully shipped containers. The first container ships left ports in 1960. Ten years later ships transporting containers were also being used in Europe.

The first containers were constructed to meet American norms, which are difficult to adapt for Europe as well as other countries. Thus, the International Organization for Standardization (ISO) standards were invented. According to ISO standards, container lengths are set to 10 feet, 20 feet, 30 feet and 40

feet, while widths are fixed at 8 feet and heights are set to 8 feet and 8 feet, 6 inches. The majority of produced containers adheres to ISO standards. In particular, 20-foot and 40-foot containers are widely spread all over the world. As stowage factors increase for most goods, so-called *Jumbo* containers that exceed sizes defined by ISO standards (45-foot and 48-foot) have been used more frequently in recent years. In the entire United States (US) it is additionally possible to transport 53-foot containers, while some states in the US even allow transportation of 57-foot containers (Duken and Gesamtverband der Deutschen Versicherungswirtschaft [2002-2016]). The length of a container is measured in twenty-foot equivalent units (TEU). Apart from the size, containers can be classified based on cargo and the different requirements of cargo. Examples are constituted by dry (normal) and special (e.g., refrigerated and hazardous) containers; the latter container type can only be handled by specially equipped container vehicles (Chung et al. [2007]). Additionally, some countries define limitations on the container's weight for container transportation. While transportation of one fully loaded 20-foot or 40-foot container is permitted for most countries, transportation of two fully loaded 20-foot containers exceeding the limitation of 26 tons is prohibited for standard vehicles. As Popović et al. [2012], Vidović et al. [2011, 2012] state, vehicles that are used in the US, Australia, Canada, Finland and Sweden offer the possibility to transport two fully loaded 20-foot containers. In contrast, EU regulations only permit certain types of vehicles (*modular concept vehicles*) to transport two fully loaded 20-foot containers by using a special combined chassis. As a result, the issue of efficiently building routes to transport containers hardly depends on the provided technical solutions.

1.1.2 Maritime Container Transportation

The portion of maritime transportation in world trade varies between 90% and 95%. In the past decade maritime traffic grew about 3% and container shipping grew about 9%.

For more than two decades container shipping constitutes the most strongly expanding shipping market. In recent years the usage of containers for intercontinental maritime transportation has been dramatically increased (Günther and Kim [2006]). Figure 1.2 depicts the port container traffic, i.e. the flow of (empty and fully loaded) containers in million TEU between the different means of transportation that are used at sea and on land. Altogether, four graphs are shown: the world's port container traffic is colored blue, the regions of East Asia and the Pacific are colored red, Europe's together with Central Asia's port container traffic is colored brown, and North America is colored

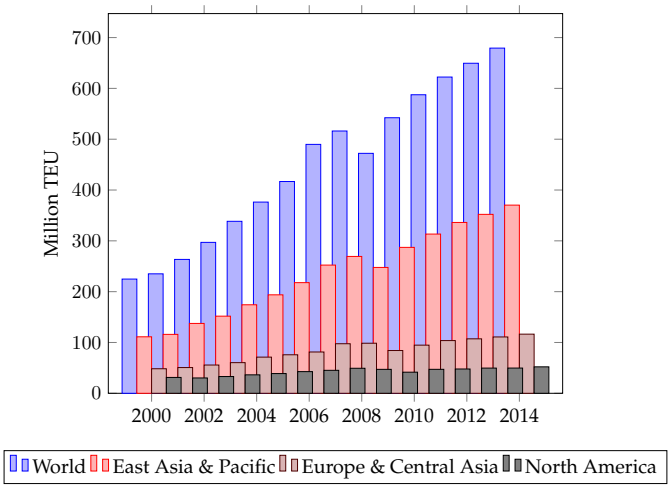


Figure 1.2: Container port traffic of the world and some regions. Data from http://data.worldbank.org, (accessed on 20th September 2016).

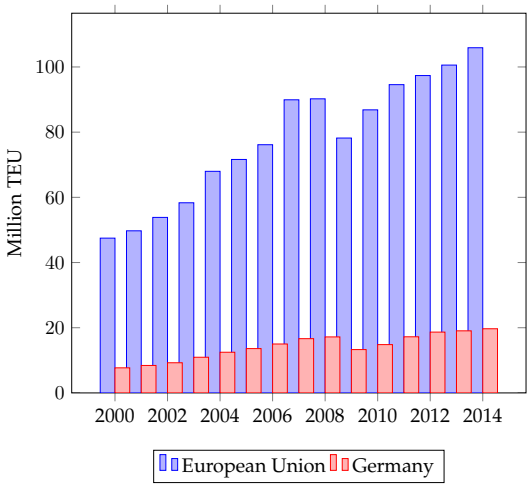


Figure 1.3: Container port traffic of the European Union and Germany. Data from http://data.worldbank.org, (accessed on 20th September 2016).

gray. In 2005, 376 million TEU were handled in harbors worldwide. The total throughput of the world in 2007 was 490 million TEU. This number rose to 679 million TEU in 2014. This trend continues; Drewry and Ocean Shipping Consultants (OSC) as well as the Bremen Institute of Shipping Economics and Logistics (ISL) forecast a doubling of the total world container turnover by 2020 compared to 2010. Günther and Kim [2006] expect a further continuous increase especially between Asia and Europe in the upcoming years. From 2000 to 2014, container port traffic from Europe and Central Asia rose from 48 million TEU to 116 million TEU. Figure 1.3 shows a similar graph development of the container port traffic of the European Union (EU) and Germany.

1.2 Intermodal Freight Transportation

Intermodal freight transportation refers to the transportation of freight in a container or vehicle by at least two different means of transportation (*modes*, e.g., rail, road, deep sea vessel, barge) in a single transportation chain. Freight is packed into loading units that do not change on their ways from senders to receivers, i.e. no handling of the freight itself is performed while it is transported and transshipped between different transportation modes.

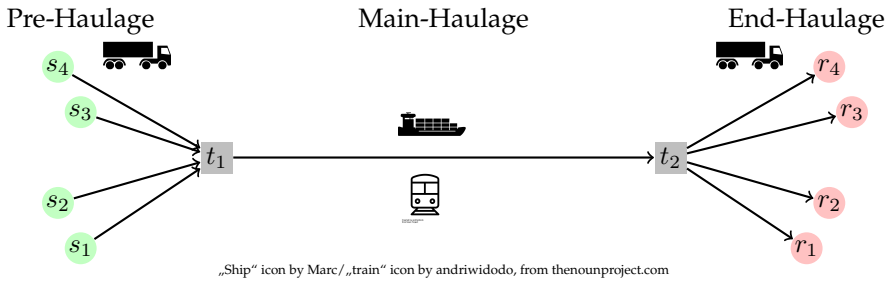


Figure 1.4: Typical transportation chain (own representation Funke and Kopfer [2014] based on Macharis and Bontekoning [2004]).

Figure 1.4 shows a typical transportation chain of a flow of goods. Three different sections are depicted, each of them being operated by a distinct transportation mode. In the first section (*pre-haulage*) containerized cargo is transported from actual customers (*senders*) s_1, s_2, s_3, s_4 to terminals by truck. The second section (*main-haulage*) consists of container transportation between terminals t_1, t_2 that is mostly carried out by barge, deep sea shipping or rail. The third section (*end-haulage*) implies container transportation from terminals to customers (*receivers*) r_1, r_2, r_3, r_4 by truck.

1.2.1 Container Hinterland Drayage

Figure 1.5 illustrates the locations and the tasks of pre- and end-haulage operations, which are proceeded in a hinterland region. The main tasks are the allocation of containers and the construction of routes for trucks to move containers. Containers have to be transported between terminals t, senders s_1, s_2, receivers r_1, r_2, r_3 and depots d. The transportation by truck constitutes one

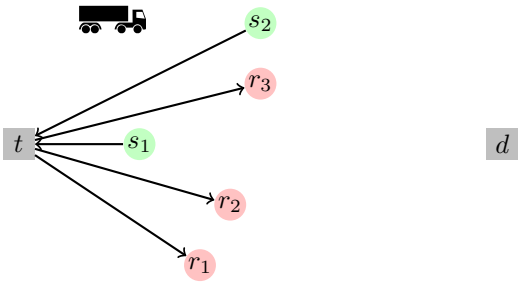


Figure 1.5: Drayage operations in an inland region of a port.

of the most important transportation modes in seaport hinterland regions. By now, transportation by truck offers the only possibility for comprehensive *door-to-door services* to customers' locations (Hildebrand [2008]). Road transportation continuously grows. Apart from the costs of drivers, the maintenance of trucks and the rising fuel prices, the costs of congested roads and the delays caused by traffic jam, the rising costs of emissions standards, toll charges and the additional costs of the new regulation on driving times and rest periods have to be considered (Hildebrand [2008]).

Pre-haulage Operations and End-haulage Operations

Pre-haulage operations include the supply of empty containers at senders' locations and the subsequent transportation of fully loaded containers from senders' locations to terminals. Empty containers can be obtained from different locations like depots, terminals and receivers' locations. It is part of the decision making process to determine the locations where empty containers can be obtained from. End-haulage operations involve the transportation of fully loaded containers from terminals to receivers' locations and the subsequent collection of empty containers at receivers' locations. Empty containers can be delivered to different locations like depots, terminals and senders' locations. Again, it is part of the decision making process to determine the locations where empty containers can be delivered to.

1.2.2 Actors

Several actors are involved in container hinterland transportation. Different actors are faced with different challenges and operations.

Carriers

Transportation of cargo is performed by a *carrier* that is engaged due to the customers' transportation demand. A container transportation request can be served in one of two ways. If the carrier owns suitable resources (e.g. ships, trucks), then the carrier is allowed to serve the entire container transportation request by itself. Another possibility is to let the carrier assigning parts of the container transportation request to hinterland and sea *transport operators*. In hinterland transportation *trucking companies* are responsible for moving containers between customers and terminals or vice versa; in sea transportation *shipping companies* are responsible for moving containers between terminals. Often the carrier is represented by one of the transport operators and instructs trusted transport operators with required transportations. Since the longest distances are crossed in the main-haulage, most of the times the shipping company organizes the entire transportation of containers between different hinterland regions (Hildebrand [2008], Sterzik [2013]). The term *carrier haulage* refers to one transport operator organizing the entire transportation for a sender. If an autonomous carrier or receiver organizes the entire transportation chain, then this service is called *merchant haulage* (Sterzik [2013], Veenstra [2005]).

Container Depots

In container depots containers are stored, repaired, maintained and cleaned. Moreover, container depots serve as transshipment points. In order to obtain a fluent hinterland service, which allows an efficient distribution of empty containers over the entire region, container depots have to be located at central positions in the hinterland region. However, placing depots at central locations stands in contrast to locating depots next to the home bases of shipping companies, which want to control their containers (Sterzik [2013], Veenstra [2005]).

Ports and Container Terminals

Ports and container terminals represent a bottleneck of the global container transportation chain. Ports have to handle mega-vessels that are able to transport up to 21,100 TEU (Merk et al. [2015]). As a result, ports have to build greater accesses for vessels. Furthermore, there is a need for efficient technologies to handle the turnover of cargo at ports (Hildebrand [2008]). Conse-

quently, container ports have to invest heavily to meet the stringent demands for obtaining faster service and higher quality. The operations at container terminals include stacking and pre-marshalling containers and transporting containers between land and sea. Container terminals are forced to reduce costs and improve efficiency to compete with other terminals. At the same time, container terminals have to meet strict instructions given by shipping companies that ask for adherence of delivery dates and promised handling times (Stahlbock and Voß [2008]). In order to cope with the increasing container turnover (refer e.g. to Figure 1.2), the capacity at ports and terminals has to be extended not only, but also the efficiency of hinterland connections has to be increased.

Trucking Companies

An intermodal terminal is used by twelve and more trucking companies. Trucking companies are faced with several challenges. Containerized cargo has to be transported by truck in a cost-effective manner. The majority of delivery is known in advance, which results in sequencing and scheduling problems between senders, receivers and at least one terminal, thereby meeting restrictions like time windows at customers' locations. Trucking companies have to give realistic prognoses on the length of a working day including break times and driving durations. The prognoses have to consider assumptions on a few short-term deliveries as well. Due to traffic and delay at terminals some deliveries might need to be reassigned. In addition, trucking companies have to provide empty trailers/containers, which are used to transport goods. Consequently, terminals store a sufficient large number of empty trailers/containers. Within one day, empty trailers/containers are scattered over the whole hinterland area resulting in unproductive movements being unavoidable.

Container Owners

The set of container owners mainly decomposes into *container leasing companies* (around 41% of the total container volume) and shipping companies (around 59% of the total container volume) (Sterzik [2013]). Only a small amount of containers is owned by depots or trucking companies. One of the main reasons for the introduction of container leasing companies is to overcome the different requirements of empty containers caused by imbalances in trade in different regions. In order to transport cargo, the carrier either uses its own available containers or hires containers from leasing companies and returns them after the cargo transportation requests have been fulfilled. Leasing contracts are distinguished into *short-term* and *long-term contracts*. Most of the times shipping

companies have a sudden demand for containers, and therefore, enter into short-term contracts. Containers that are hired by a long-term contract stay at shipping companies for months and years (Sterzik [2013]). Shipping companies and container leasing companies are pursuing conflicting objectives; in contrast to shipping companies, which want to minimize container transportation and container handling costs, container leasing companies want to maximize the profit through the leasing of containers.

1.2.3 Empty Container Repositioning

The most of the flows of containerized cargo is asymmetric. Therefore, it is necessary to reposition empty containers, which results in unproductive movements. The repositioning of empty containers does not generate any revenue, unproductive movements are associated with container handling and transshipment costs instead. Furthermore, empty containers have to be stored, maintained, cleaned and repaired. Consequently, the repositioning of empty containers marks one of the ongoing issues for operating actors of sea and land transportation (Sterzik [2013]). Due to its expensiveness, the repositioning of empty containers has become one of the most important problems in the shipping industry over the last years. The need for unproductive movements appears on global and regional levels: between export- and import-oriented areas, and between senders and receivers acting in the same hinterland region.

Imbalances in trade cause regions having different supply and demand for empty containers, for instance the *outbound* flows from Asia to Europe and North America were more than twice as high as the *inbound* flows of these regions in 2010 (Sterzik [2013]). Export-oriented areas (like the Far East) export more cargo to import-oriented regions (like Western regions) than vice versa. As a result, export-oriented areas have to cope with a lack of empty containers, while import-oriented areas are confronted with an overrun of empty containers. Unproductive movements represent a significant cost driver within the entire container transportation chain. Drewry Shipping Consultants of London estimate the share of empty containers that are repositioned at sea to be around 20% of all maritime container movements (Boile et al. [2006], Braekers et al. [2011]). Estimations of the rate of repositioned empty containers for land transportation range from 40% to 50% (Braekers et al. [2011], Branch [2006], Crainic et al. [1993], Konings and Thijs [2001]).

In order to reduce unproductive movements in hinterland regions, the balance of empty containers needed by senders and provided by receivers has to be improved. Jula et al. [2006] analyze two different strategies for empty container repositioning in hinterland regions. The first strategy (*depot-direct*

strategy) tries to relief intermodal terminals that store empty containers. Therefore, several off-dock container depots are introduced at different places. The implementation of further capacity in addtion to intermodal terminals should minimize the distances that are caused by rearranging empty containers. In the second strategy (*street-turn strategy*) empty containers that become available at receivers' locations are directly moved to senders' locations where they are needed next. The street-turn strategy minimizes distances that are caused by taking a detour via a depot for container stowage operations. The great cost saving potential of street-turns stands highly in contrast to the small amount of empty containers that is directly transferred between customers' locations in real-world applications (Sterzik [2013]). Limiting factors for street-turns are for instance time windows at customer's locations that do not match, expiring contracts of leased containers and different container types.

1.3 Objectives and Structure of the Thesis

The objective of the present thesis is to pursue a further development in the research area on container hinterland transportation by making a major contribution regarding the transportation of 20-foot and 40-foot containers. To achieve this, a widely studied drayage problem is investigated in new terms by considering 20-foot and 40-foot containers, resulting in the definition of the *multi-size Inland Container Transportation Problem (mICTP)*. There is extensive literature on the transportation of 40-foot containers by truck available (refer to Chapter 3). However, the simultaneous transportation of containers having different sizes constitutes an application that is often neglected in related literature sources[1]. In this thesis, solution methodologies are designed to increase the transportation efficiency of trucking companies operating in the hinterland region of a seaport. This goal is achieved by reducing transportation costs. Trucking companies are concerned with the movement of containerized cargo as well as the provision of empty containers, leading to route construction problems for trucks as well as assignment problems of empty containers to transportation requests for cargo. A further consideration that is rarely investigated in literature sources is the separation of trucks and containers for container handling operations (e.g., loading and unloading containers)[2]. During the time needed to (un-)load a container the truck is free to carry out other tasks.

[1] See Section 3.4 for further details on literature sources investigating drayage problems for containers having different sizes.

[2] See Section 3.3.2 for a review on literature sources investigating drayage problems, in which trucks and containers are permitted to separate.

The present thesis consists of eight chapters and one attachment.

Chapter 1 introduces the topic, actors, challenges and cost drivers that are involved in the research field of container hinterland transportation. Furthermore, the motivation of the thesis is outlined and an overview of the thesis is given.

After the introduction, Chapter 2 moves on with defining and discussing mathematical and logistical problem definitions that are needed for a deeper understanding of the thesis. Different heuristic solution procedures and exact formulations that are applied in literature sources are presented; these approaches compute solutions belonging to problem classes similar to the class containing the mICTP.

An extensive literature overview is given in Chapter 3. As can be seen from this chapter, several scientific papers addressing container hinterland transportation problems can be found in literature. Chapter 3 classifies the mICTP and points out the novelty of the considered problem definition.

As Chapter 5 and Chapter 6 both introduce solution methodologies for the mICTP, the formal definition of the mICTP is separately given in Chapter 4.

Chapter 5 presents an exact mathematical programming approach to compute solutions to the mICTP. In a later chapter it will be shown that the implementation of the exact approach together with a commercial solver is able to compute solutions to instances consisting of around eleven transportation requests and ten trucks.

Chapter 6 presents a heuristic approach to compute solutions to the mICTP. This chapter outlines a relevant high-quality approach that can be applied to instances of the mICTP arising in real-world scenarios.

The approaches of Chapters 5 and 6 are implemented, and these implementations are evaluated in Chapter 7. In the evaluation two points are considered in particular: the solution quality and the practicability of the approaches. The benchmark instances in this chapter contain randomly generated instances as well as instances that are taken from literature sources.

Chapter 8 concludes the thesis and gives an outlook of extensions that could improve both the definition and solution approaches for the mICTP.

Attachment A shows detailed and summarized tables containing computational results that are not mentioned in Chapter 7.

Chapter 2

Basic Definitions

The following definitions and notations are needed by the remainder of this thesis. Section 2.1 introduces some basic Combinatorial Optimization (CO) problems and useful representations of these problems, like linear and integer programs (Section 2.1.1) and graphs (Section 2.1.2). Sections 2.1.3 and 2.1.4 give examples of CO problems that are required by Chapters 3, 5 and 6. The structure of the problem introduction in these sections is as follows. Whenever a problem P is newly introduced, it will be explained whether or not an algorithm, which computes an optimum solution to P in polynomial running time, exists. If such an algorithm exists, then the running time $\mathcal{O}(f)$ of the fastest known algorithm f for P will be stated. On the other hand, if no such polynomial time algorithm for P is known, then it will be stated, whether or not P is $\mathcal{NP}$ (nondeterministic polynomial time)-hard. Furthermore, if it is required by later chapters, linear and integer programming formulations of P will be presented. Section 2.2 comprises algorithms for CO problems. In this chapter we primarily follow the notations of Korte and Vygen [2008].

2.1 Combinatorial Optimization Problems

According to Blum and Roli [2003] many optimization problems aim in computing the best configuration of a set of variables in order to achieve some predefined goals. Typically, these problems can be distinguished by their solutions, namely solutions that are allowed to consist of real-valued variables and solutions that have to consist of discrete variables. CO problems are included in the second class.

2.1.1 Linear, Integer and Mixed Integer Programming

The objective of Operations Research (OR) is the development and usage of mathematical procedures to support decision-making processes (Alisch et al. [2013]). A major area of OR comprises the transformation of problems arising in practice to abstract optimization problems. As Williams [2013] states, model building in mathematical programming covers a wide range of applications in many diverse areas: physical laws, technical relationships, company internal structures or the presented container transport problem can be represented in mathematical relationships by using (in-)equalities and logical dependencies. Linear programming is a widely used procedure in the field of OR.

Definition 2.1 (Linear, Integer and Mixed Integer Programming)**.** Given a matrix $A \in \mathbb{R}^{m \times n}$, a column vector $b \in \mathbb{R}^m$ and a row vector $c \in \mathbb{R}^n$, **Linear Programming** asks for a column vector $x \in \mathbb{R}^n$ maximizing cx such that $Ax \leq b$, or the decision whether the solution space $\{ x \in \mathbb{R}^n \mid Ax \leq b \}$ is either empty (the problem is **infeasible**) or the value maximizing cx converges to infinity for x in the solution space (the problem is **unbounded**). An instance of Linear Programming is called **Linear Program (LP)**. The elements of the solution space are called **feasible solutions**. A feasible solution attaining the maximum is an **optimum solution**. The elements of x are called **decision variables**. If the elements of x are restricted to be integral, then the problem is called **Integer Programming** and the instance is called **Integer Program (IP)**. If only a sub-set of x is restricted to obtain integral values, while the remaining values of x are allowed to obtain continuous values, then the problem is called **Mixed Integer Programming** and the instance is called **Mixed Integer Program (MIP)**.

2.1.2 Graphs

The illustration of CO instances in the form of a graph often improves knowledge and comprehension of the underlying problem formulation.

Definition 2.2. Let $k \in \mathbb{N}$ be a natural number. We denote by $[k]$ the set of natural numbers less than or equal to k, i.e. $[k] := \{1, 2, \ldots, k\}$.

Definition 2.3 (Partition)**.** A **partition** of a set V is a family M of sub-sets of V, such that the elements of M are pairwise disjoint, i.e.:

- $\forall m \in M : m \subseteq V$
- $\forall m, m' \in M : m \cap m' = \emptyset$

The symbol $\dot{\cup}$ denotes a disjoint union of sets.

Definition 2.4 (Graphs, Nodes, Arcs)**.** An **undirected graph** G is a pair of finite sets $G = (V, A)$, where V is a nonempty set and E is a set comprising pairs of two different elements of V, i.e. it holds for $e \in E$: $e \subseteq V$ with $|e| = 2$. A directed graph or **digraph** G is a pair of finite sets $G = (V, A)$, where again V is a nonempty set and E comprises ordered pairs of two different elements of V, i.e. it holds for $e \in E$: $e = (v, w) \in V \times V$ with $v \neq w$. A graph is either undirected or directed. The elements of V are called **nodes** and the elements of A are called **arcs**. A **sub-graph** $G' = (V', A')$ of a graph $G = (V, A)$ is a graph with $V' \subseteq V$ and $A' \subseteq A$. Let $e = (v, w) \in A$ be an arc, the nodes v and w are called the **end-points** of e; the node v is called the **tail** and the node w is called the **head** of e. Furthermore, the nodes v and w are **incident** with e, while v and w are **adjacent**. Let $G = (V, A)$ be a digraph and $v \in V$ be a node. We denote by $\delta_v^+ := \{\, (v, w) \,|\, (v, w) \in A \,\} \subseteq A$ the set of out-going and by $\delta_v^- := \{\, (w, v) \,|\, (w, v) \in A \,\} \subseteq A$ the set of in-going arcs. An undirected graph is called **complete**, if all nodes are adjacent with each other.

Unless otherwise stated, a graph is denoted by the pair $G = (V, A)$. The following mappings are often used in the present thesis.

Definition 2.5. Let $e := (v, w)/\{v, w\} \in A$ be an arc. The **cost** of e is denoted by $c_{vw} \in \mathbb{R}$ and its **capacity** is denoted by $u_{vw} \in \mathbb{R}_+$. A graph together with a cost function is called **weighted**. Let $v \in V$ be a node. The **balance** of v is denoted by $b_v \in \mathbb{R}$. In the case of more than one ($1 < \dim \in \mathbb{N}$) different types of commodities, we denote by the vector $b_v^{(k)}$ for $k \in [\dim]$ the balance of v .

Graphs may have several different properties.

Definition 2.6 (Path, Cycle)**.** Let $1 \leq n$ be a natural number. A **path** from v_1 to v_n (or v_1-v_n-path) is a graph $P := (\{v_1, v_2, \ldots, v_n\}, \{e_1, e_2, \ldots, e_{n-1}\})$, such that $e_i := (v_i, v_{i+1})/\{v_i, v_{i+1}\}$ for all $1 \leq i \leq n-1$ and $v_i \neq v_j$ for all $1 \leq i < j \leq n$. A graph $C := (\{v_1, v_2, \ldots, v_n\}, \{e_1, e_2, \ldots, e_n\})$ is a **cycle**, if $v_1 = v_n$ and the sub-graph $C' := (\{v_1, v_2, \ldots, v_{n-1}\}, \{e_1, e_2, \ldots, e_{n-1}\}) \subset C$ is a path. A graph G is called **acyclic**, if G does not contain any sub-graph that is a cycle. In digraphs the term **directed path/cycle** also refers to a path/cycle.

Definition 2.7 (Connected, Connected Components)**.** An undirected graph $G = (V, A)$ is called **connected**, if there is a v-w-path for all pairs of nodes $v, w \in V$. The maximal connected sub-graphs of G are called **connected components** of G. A digraph G is called **connected**, if the underlying undirected graph is connected. The maximal connected sub-graphs of the underlying undirected graph of G are called the **connected components** of G.

A more strict definition is given for digraphs.

Definition 2.8 (Strongly Connected Components, Condensation)**.** A digraph $G = (V, A)$ is called **strongly connected**, if there are a v-w-path and a w-v-path for all pairs of nodes $v, w \in V$. The maximal strongly connected sub-graphs of G are called **strongly connected components** of G. The **condensation** of a digraph $G = (V, A)$ is a digraph $G' = (V', A')$ with:

$$\begin{aligned} V' &:= \{\, C \subseteq G \mid C \text{ is a strongly connected component of } G \,\} \\ A' &:= \{\, (C_v, C_w) \subseteq G \times G \mid C_v \neq C_w : \exists v \in C_v, \exists v \in C_w : (v, w) \in A \,\} \end{aligned} \tag{2.1}$$

Remark 2.9. Strongly connected components or the condensation of a digraph $G = (V, A)$ can be obtained in $\mathcal{O}(|V| + |A|)$ by using $|V|$ times *Depth-First Search* (König [1936]) or *Breadth-First Search* (Moore [1959]).

Definition 2.10 (Transitive Closure)**.** A digraph $G = (V, A)$ is called **transitive**, if the following relation holds:

$$\forall u, v, w \in V : (u, v) \in A \text{ and } (v, w) \in A \implies (u, w) \in A \tag{2.2}$$

The **transitive closure**[1] of a digraph G, is the digraph $G' = (V, A')$ with $A' := \{\, (v, w) \in V \times V \mid G \text{ contains a } v\text{-}w\text{-path} \,\}$.

In Section 6.3.1 the construction of lower bounds on the number of trucks needed to serve all requests of an mICTP instance is proposed. In order to construct this lower bound, the definition of bipartite graphs is required.

Definition 2.11 (Bipartite Graph)**.** Let $G = (V, A)$ be an (un-)directed graph. A **bipartition** of G is a partition of the node set $V = X \dot\cup Y$, such that the head and the tail of each arc are contained in different sets, i.e. $\forall e \in A : e \cap X \neq \emptyset$ and $e \cap Y \neq \emptyset$. We denote bipartite graphs by $G = (X \dot\cup Y, A)$. A bipartite graph is called **complete**, if each pair of nodes $v \in X, w \in Y$ is adjacent in G.

Definition 2.12 (Vertex Cover)**.** Let G be an (un-)directed graph. A **vertex cover** in G is a set of nodes $VC \subseteq V$, such that each arc of G is incident to at least one node in VC.

Definition 2.13 (Matching)**.** A **matching** in an undirected graph $G = (V, A)$ is a set of arcs $M \subseteq A$, such that the arcs in M are pairwise disjoint, i.e.:

$$\forall m, m' \in M : m \cap m' = \emptyset$$

A matching is **maximal**, if M will no longer be a matching by adding an arc $e \in A$ to M. A matching is **perfect**, if each node $v \in V$ is incident with an arc in M.

[1] Remark 2.27 shows a construction of the transitive closure of a digraph.

Karp [1972] proofs that computing the minimum cardinality of a vertex cover is in general $\mathcal{NP}$-hard. However, this attribute does not apply to bipartite graphs:

Theorem 2.14 (König [1931])**.** *Let G be a bipartite graph. The maximum cardinality of a matching*[2] *in G equals the minimum cardinality of a vertex cover in G.*

2.1.3 Flow Problems

Two sub-problems constitute the mICTP: empty container assignment and route construction. Therefore, this section introduces a bunch of flow problems, which are required by the assignment problem of empty containers, while Section 2.1.4 introduces routing and scheduling problems, which are required in order to construct routes.

ASSIGNMENT PROBLEM

Instance: A weighted bipartite graph $G = (X \dot{\cup} Y, A)$, where X and Y have the same cardinality ($|X| = |Y|$), together with costs $c : A \to \mathbb{R}$.

Task: Find a perfect matching M minimizing the total cost $c(M) := \sum_{e \in M} c_e$, or decide that no perfect matching in G exists.

An IP formulation of the Assignment Problem (AP) is stated in the following. A binary decision variable δ_{vw} is introduced for each arc $(v, w) \in A$:

$$\delta_{vw} := \begin{cases} 1, & \text{if node } v \text{ is assigned to node } w \\ 0, & \text{otherwise} \end{cases}$$

The AP can be formulated as follows:

$$\min \sum_{\{v,w\} \in A} c_{vw} \delta_{vw} \tag{2.3}$$

such that

$$\sum_{v \in X} \delta_{vw} = 1 \qquad \forall w \in Y \tag{2.4}$$

$$\sum_{w \in Y} \delta_{vw} = 1 \qquad \forall v \in X \tag{2.5}$$

$$\delta_{vw} \in \{0, 1\} \qquad \forall \{v, w\} \in A \tag{2.6}$$

The Objective function 2.3 minimizes the total cost of the assignments. Constraints 2.4 and 2.5 assign exactly one element of the first bipartition to the second bipartition and vice versa. The domains of the decision variables are specified by Constraints 2.6.

[2] A polynomial time algorithm for the computation of a maximum cardinality matching in bipartite graphs is given by Remark 2.18.

Remark 2.15. Because of its special structure, the AP can be solved in polynomial time, e.g. by the Hungarian Method (Kuhn [1955]) in $\mathcal{O}(|X \dot{\cup} Y|^3)$.

While the AP can efficiently be applied to assign 40-foot containers, the assignment problem of 20-foot containers requires more complex problem definitions.

Definition 2.16 (s-t-Flow)**.** Let $G = (V, A)$ be a digraph with capacities $u : A \to \mathbb{R}^+$. For two elements $s, t \in V$, a **s-t-flow** in G is a function $f : A \to \mathbb{R}^+$ satisfying the **capacity constraints**

$$\forall e \in A : f(e) \leq u_e \tag{2.7}$$

and the **flow conservation constraints**

$$\forall v \in V \setminus \{s, t\} : \sum_{e \in \delta_v^+} f(e) - \sum_{e \in \delta_v^-} f(e) = 0 \tag{2.8}$$

together with

$$\sum_{e \in \delta_s^+} f(e) - \sum_{e \in \delta_s^-} f(e) \geq 0, \qquad \sum_{e \in \delta_t^+} f(e) - \sum_{e \in \delta_t^-} f(e) \leq 0 \tag{2.9}$$

MAXIMUM FLOW PROBLEM

Instance: A digraph $G = (V, A)$, capacities $u : A \to \mathbb{R}^+$ and two nodes $s, t \in V$.

Task: Find a s-t-flow f maximizing the value $\sum_{e \in \delta_s^+} f(e) - \sum_{e \in \delta_s^-} f(e)$.

With the help of a continuous decision variable f_e for each arc $e \in A$ that represents the s-t-flow, a LP formulation of the Maximum Flow Problem can be stated as follows:

$$\max \sum_{e \in \delta_s^+} f(e) - \sum_{e \in \delta_s^-} f(e) \tag{2.10}$$

such that

$$\sum_{e \in \delta_v^+} f(e) - \sum_{e \in \delta_v^-} f(e) = 0 \qquad \forall v \in V \setminus \{s, t\} \tag{2.11}$$

$$f_e \leq u_e \qquad \forall e \in A \tag{2.12}$$

$$f_e \in \mathbb{R}^+ \qquad \forall e \in A \tag{2.13}$$

The Objective function 2.10 maximizes the total amount of flow traversing from s to t. Flow conservation constraints are represented by Constraints 2.11 and capacity constraints are represented by Constraints 2.12. The domains of the decision variables are specified by Constraints 2.13.

Remark 2.17. Dantzig and Fulkerson [1956] proof that integral instances (capacities $u : A \to \mathbb{N}$) of the Maximum Flow Problem have got integral optimum solutions, which can be obtained in polynomial running time. One of the fastest known algorithms is able to compute optimum solutions to the Maximum Flow Problem in $\mathcal{O}(|V| \cdot |A| \cdot \log_{2+\frac{|A|}{|V| \cdot \log |V|}} |V|))$ (King et al. [1992]).

Remark 2.18. The maximum cardinality matching in a bipartite graph $G = (V, A)$ can be solved as a Maximum (s-t-)Flow Problem with uniform capacities $u \equiv 1$ in a digraph $G' = (V', A')$ that is defined as follows:

$$\begin{aligned} V' &:= V \cup \{s, t\} \\ A' &:= A \cup \{\, \{s, v\}, \{w, t\} \mid (v, w) \in A \,\} \end{aligned} \tag{2.14}$$

Definition 2.19 (b-Flow)**.** Let $G = (V, A)$ be a digraph with capacities $u : A \to \mathbb{R}^+$ and balance values $b : V \to \mathbb{R}$ with $\sum_{v \in V} b_v = 0$. A **b-flow** in G is a function $f : A \to \mathbb{R}^+$ satisfying the capacity constraints (Constraints 2.7) and the **flow balance conservation constraints**

$$\forall v \in V : \sum_{e \in \delta_v^+} f(e) - \sum_{e \in \delta_v^-} f(e) = b_v \tag{2.15}$$

MINIMUM COST FLOW PROBLEM

Instance: A weighted digraph $G = (V, A)$, capacities $u : A \to \mathbb{R}^+$, balance values $b : V \to \mathbb{R}$ with $\sum_{v \in V} b(v) = 0$ and costs $c : A \to \mathbb{R}$.

Task: Find a b-flow f minimizing the total cost $c(f) := \sum_{e \in A} c_e \cdot f(e)$, or decide that no b-flow in G exists.

We obtain a LP formulation of the Minimum Cost Flow Problem (MCFP) from the Maximum Flow formulation by replacing the Objective function 2.10 by

$$\min \sum_{e \in A} c_e f_e \tag{2.16}$$

and by replacing the flow conservation Constraints 2.11 by flow balance conservation constraints

$$\sum_{e \in \delta_v^+} f_e - \sum_{e \in \delta_v^-} f_e = b_v \qquad \forall v \in V \tag{2.17}$$

Remark 2.20. Similar to the formulation of the AP and the Maximum Flow Problem, the LP formulation of the MCFP has got a special structure. Let $G = (V, A)$ be an instance of the MCFP. The existence of an optimum b-flow in G that is integral ($f_e \in \mathbb{N}$ for all arcs $e \in A$) follows for instances having integral balance values ($b : V \to \mathbb{N}$) and integral capacities ($u : A \to \mathbb{N}$). The optimum

integral b-flow can be computed in polynomial time, e.g. by using the fastest known polynomial algorithm for the general uncapacitated MCFP instances in $\mathcal{O}(|V| \cdot \log |A| \cdot (|A| + |V| \cdot \log |V|))$ time (Orlin [1988]).

The Transportation Problem (TP) (Hitchcock [1941]) states a further famous problem definition that derives from the MCFP on bipartite digraphs without capacities (Constraints 2.7):

TRANSPORTATION PROBLEM

Instance: A bipartite digraph $G = (X \dot\cup Y, A)$ with $A \subseteq X \times Y$, balance values $b : V \to \mathbb{R}$ with $b_v \geq 0$ (supplies) for $v \in X$, $b_v \leq 0$ (demands) for $v \in Y$, $\sum_{v \in V} b(v) = 0$ and costs $c : A \to \mathbb{R}$.

Task: Find a function (flow) $f : A \to \mathbb{R}^+$ satisfying the flow balance constraints (Constraints 2.15) and minimizing the total cost $c(f) := \sum_{e \in A} c_e \cdot f(e)$, or decide that no such function f in G exists.

A polynomial time algorithm for the computation of optimum integral solutions is known to all of the flow problems that have been mentioned so far. However, the existence of such algorithms to the remaining problem definitions has not been proved up until now.

Definition 2.21 (Multicommodity-Flow)**.** Let $2 \leq \dim \in \mathbb{N}$ be a natural number and let $G = (V, A)$ be a digraph with capacities $u : A \to \mathbb{R}^+$ and multidimensional balance values $b^{(k)} : V \to \mathbb{R}$ with $\sum_{v \in V} b_v^{(k)} = 0$ for $k \in [\dim]$. A **multicommodity-flow** is a function $f^{(k)} : A \to \mathbb{R}^+$ for $k \in [\dim]$ satisfying the **combined capacity constraints**

$$\forall e \in A : \sum_{k \in [\dim]} f_e^{(k)} \leq u_e \tag{2.18}$$

and the flow balance conservation constraints (Constraints 2.15) for each commodity $k \in [\dim]$.

MULTICOMMODITY FLOW PROBLEM

Instance: A digraph $G = (V, A)$, capacities $u : A \to \mathbb{R}^+$, a natural number $\dim \in \mathbb{N}$, multidimensional balance values $b^{(k)} : V \to \mathbb{R}$, $k \in [\dim]$ with $\sum_{v \in V} b_v^{(k)} = 0$ for all $k \in [\dim]$ and costs $c : A \to \mathbb{R}$.

Task: Find a multicommodity-flow $f^{(k)}$ minimizing the total cost $c(f) := \sum_{e \in A} \sum_{k \in [\dim]} c_e \cdot f^{(k)}(e)$, or decide that no multicommodity-flow in G exists.

A continuous decision variable f_e^k is introduced for each arc $e \in A$ and for each dimension $k \in [\dim]$ in order to give a LP formulation of the Multicommodity Flow Problem (MCP):

$$\min \sum_{e \in A} \sum_{k \in [\dim]} c_e f_e^{(k)} \tag{2.19}$$

such that

$$\sum_{e \in \delta_v^+} f_e^{(k)} - \sum_{e \in \delta_v^-} f_e^{(k)} = b_v^{(k)} \qquad \forall v \in V, \forall k \in [\dim] \tag{2.20}$$

$$\sum_{k \in [\dim]} f_e^{(k)} \leq u_e \qquad \forall e \in A \tag{2.21}$$

$$f_e^{(k)} \in \mathbb{R}^+ \qquad \forall e \in A, \forall k \in [\dim] \tag{2.22}$$

The Objective function 2.19 minimizes the total cost of the flow. Flow balance conservation constraints for each dimension are represented by Constraints 2.20 and combined capacity constraints are represented by Constraints 2.21. Constraints 2.22 define the domains of the decision variables.

Remark 2.22. Even et al. [1976] show that producing an integral multicommodity flow (i.e. replacing Constraints 2.22 by $f_e^{(k)} \in \mathbb{N}, \forall e \in A, \forall k \in [\dim]$) is $\mathcal{NP}$-hard even if there are only two commodities specified by the problem instance.

2.1.4 Routing and Scheduling Problems

The CO problems mentioned in this section are especially important for the construction of routes of the mICTP. The following chapters often use the term *distance/distance matrix* to refer to the cost mapping of a weighted graph $G = (V, A)$.

Definition 2.23. A distance matrix is called **symmetric**, if $c_{vw} = c_{wv}$ holds for all pairs of nodes $v, w \in V$; otherwise the distance matrix is called **asymmetric**.

Definition 2.24. Let $G = (V, A)$ be a weighted digraph with costs $c : A \to \mathbb{R}$. The mapping c is called **conservative**, if there is no negative cost cycle $C = (V_c, A_c)$, i.e. $\sum_{e \in A_c} c_e \geq 0$ for all cycles $C \subseteq G$.

SHORTEST PATH PROBLEM

Instance: A digraph $G = (V, A)$, conservative costs $c : A \to \mathbb{R}$ and two nodes $s, t \in V$.

Task: Find a s-t-path P minimizing the total cost $c(P) := \sum_{e \in P} c_e$, or decide that no s-t-path exists.

The Shortest Path Problem (SP) can be formulated as a MCFP with unlimited capacities by introducing balances $b : V \to \mathbb{R}$, which are defined as follows:

$$b_v := \begin{cases} 1, & v = s \\ -1, & v = t \\ 0, & \text{otherwise} \end{cases}$$

Remark 2.25. Let $G = (V, A)$ be a weighted digraph with conservative costs. The Moore-Bellman-Ford Algorithm (Bellman [1958], Ford [1956], Moore [1959]) solves the SP in $\mathcal{O}(|V| \cdot |A|)$. This is the fastest known polynomial time algorithm for the SP with conservative costs. However, if the costs are additionally restricted to be nonnegative ($c : A \to \mathbb{R}^+$), then the running time improves to $\mathcal{O}(|A| + |V| \cdot \log |V|)$ by using an implementation of the algorithm of Dijkstra [1959] (Fredman and Tarjan [1987]).

ALL PAIRS SHORTEST PATH PROBLEM

Instance: A digraph $G = (V, A)$ and conservative costs $c : A \to \mathbb{R}$.
Task: Solve the SP for all $s, t \in V$.

Observation 2.26. The fastest known algorithm for the All Pairs Shortest Path Problem (APSP) computes in $\mathcal{O}(|A| \cdot |V| + |V|^2 \log\log |V|)$ (Pettie [2004]) the optimum solution to an APSP instance $(G = (V, A), c)$.

Observation 2.27. The transitive closure of a graph G (refer to Definition 2.10) can be computed by solving the APSP in G.

While the SP and the APSP can be efficiently solved in polynomial running time, the remaining problem definitions are much more complex; up until now, the existence of polynomial time algorithms to those problems has not been proven.

Definition 2.28 (Hamiltonian Circuit)**.** A **Hamiltonian Circuit** or **tour** of a graph $G = (V, A)$ is a cycle $G' = (V, A') \subseteq G$ on the same set of nodes.

In scientific sources, the following problem has received considerable attention specifically because of its conceptual simplicity, which is in contrast to the difficulty of solving instances of this problem type in practice (Williams [2013], Hoffman et al. [2013]).

TRAVELING SALESMAN PROBLEM

Instance: A complete graph $G = (V, A)$ and costs $c : A \to \mathbb{R}$.
Task: Find a Hamiltonian circuit $G' = (V, A') \subseteq G$ minimizing the total cost $c(G') := \sum_{e \in A'} c_e$.

The Traveling Salesman Problem (TSP) problem can be formulated similar to the AP by extending Constraints 2.4 and 2.5 to the entire node set V and by introducing a binary decision variable that is defined as follows:

$$\delta_{vw} := \begin{cases} 1, & \text{if arc } (v,w) \text{ is included in the Hamiltonian Circuit} \\ 0, & \text{otherwise} \end{cases}$$

The current formulation allows solutions to comprise more than one cycle (*sub-tours*), instead of only one Hamiltonian Circuit. Several different approaches for the exclusion of sub-tours are shown by literature sources; the mathematical model of the mICTP includes the sub-tour elimination constraints of Miller et al. [1960], which can be easily modified to also implement time window constraints. The sub-tour elimination constraints of Miller et al. [1960] require a continuous decision variable m_v for each node $v \in V$ together with a special node (the depot) $d \in V$:

$$m_d = 0 \tag{2.23}$$

$$\delta_{vw} = 1 \implies m_w \geq m_v + 1 \qquad \forall (v,w) \in A \setminus \delta_d^- \tag{2.24}$$

MULTIPLE TRAVELING SALESMAN PROBLEM

Instance: A complete weighted graph $G = (V, A)$, a node $d \in V$, costs $c : A \to \mathbb{R}$ and a natural number $k \in \mathbb{N}$.

Task: Find k cycles $G_k = (V_k, A_k) \subseteq G$, such that $d \in V_k$ for all k and $V = \dot{\cup}_{i \in [k]} (V_i \setminus \{d\}) \cup \{d\}$ minimizing the total cost $\sum_{i \in [k]} \sum_{e \in A_i} c_e$.

An IP formulation of the Multiple Traveling Salesman Problem (mTSP) can be obtained by replacing the Constraints 2.4 and 2.5 for the set δ_d^+ / δ_d^+ by

$$\sum_{e \in \delta_d^+} \delta_e = \sum_{e \in \delta_d^-} \delta_e = k \tag{2.25}$$

There is a wide variety of (m)TSP problems presented in scientific literature, like time window and synchronization constraints (refer to Sections 3.1.1 and 3.1.2 for variations of the (m)TSP). The (m)TSP can be distinguished in symmetric and asymmetric problem definitions regarding to its practical application. The additional consideration of balance values for nodes and capacity limitations for routes leads to a problem that is called *Vehicle Routing Problem*.

The *Full Truckload Pickup and Delivery Problem (FTPDP)* is a *Pickup and Delivery Problem (PDP)* (see e.g. Berbeglia et al. [2007], Parragh et al. [2008a,b], Savelsbergh and Sol [1995]) with a special structure such that it can be formulated as an asymmetric mTSP (amTSP) (see also Sections 3.1 and 3.2).

Definition 2.29 (FTPDP structure)**.** Let $G = (\{d\} \cup V^+ \cup V^-, A)$ be a graph, where the ordered sets $V^+ = (v_1^+, \dots v_n^+)$ and $V^- = (v_1^-, \dots v_n^-)$ have the same cardinality n. Then G includes a **FTPDP structure**, if the arc set is defined as follows:

$$A := \left\{ (v_i^+, v_i^-) \,\middle|\, i \in \{1, \dots, n\} \right\} \cup \left\{ (v, w) \,\middle|\, v \in \{d\} \cup V^-, w \in \{d\} \cup V^+ \right\}$$

This means that each node in the set V^+/V^- is the tail/head of exactly one arc, while the remaining graph is complete.

Given this definition, the FTPDP can be formulated analogous to the mTSP:

FULL TRUCKLOAD PICKUP & DELIVERY PROBLEM

Instance: A graph $G = (\{d\} \cup V^+ \cup V^-, A)$ in a FTPDP structure, costs $c : A \to \mathbb{R}$ and a natural number $k \in \mathbb{N}$.

Task: Find k cycles $G_k = (V_k, A_k) \subseteq G$, such that $d \in V_k$ for all k and $V = \dot{\cup}_{i \in [k]} (V_i \setminus \{d\}) \cup \{d\}$ minimizing the total cost $\sum_{i \in [k]} \sum_{e \in A_i} c_e$.

Because of the FTPDP structure, each feasible solution to the FTPDP covers the arc (v_i^+, v_i^-) for all $i \in \{1, \dots, n\}$. In other words, a good is picked up at node $v_i^+ \in V^+$ and delivered to node $v_i^- \in V^-$ on the direct route.

2.1.5 More Mathematics

The following definitions on sets and properties of sets are required by Section 6.3.1. Section 6.3.1 gives a construction of lower bounds on the number of trucks needed to serve all requests of a mICTP instance.

Definition 2.30 (Chains, Antichains)**.** A **set** $R := \{r_1, r_2, \dots, r_n\}$ is a collection of distinct elements. A **partial order** over a set is a relation $\leq$ that is reflexive, antisymmetric and transitive, i.e.:

- reflexivity: $\forall r \in R : r \leq r$
- antisymmetry: $\forall r_1, r_2 \in R : r_1 \leq r_2$ and $r_2 \leq r_1 \implies r_1 = r_2$
- transitivity: $\forall r_1, r_2, r_3 \in R : r_1 \leq r_2$ and $r_2 \leq r_3 \implies r_1 \leq r_3$

A **partially ordered set (*poset*)** $(R, \leq)$ is a set together with an partial order. Two elements $r_1, r_2 \in R$ are **comparable**, if $r_1 \leq r_2$ or $r_2 \leq r_1$ holds. **Chains/antichains** are subsets of pairwise comparable/incomparable elements of R.

Theorem 2.31 (Dilworth [1950])**.** *In any finite poset the maximum size of an antichain equals the minimum number of chains into which the poset can be partitioned.*

The subsequent $\mathcal{NP}$-hard problems are mentioned in the literature review that is contained in Chapter 3:

Definition 2.32. Given an ordered set $S = (s_1, \ldots, s_n)$ of objects and a class $\mathcal{S}$ of sub-sets of S together with a cost function $c : \mathcal{S} \to \mathbb{R}$. The **Set Covering Problem** asks for a selection $\mathcal{S}'$ of sub-sets of $\mathcal{S}$, such that each element of S is contained in at least one element of $\mathcal{S}'$, thereby minimizing the total cost $\sum_{i \in \mathcal{S}'} c(i)$. The **Set Partitioning Problem** asks (if existent) for a partition of S consisting of elements of $\mathcal{S}$ minimizing the total cost.

2.2 Algorithms

This section provides a general overview of algorithms for CO problems; we mainly follow the notations of Blum and Roli [2003] in the following. Algorithms for CO problems can be categorized into *complete* and *approximate algorithms*.

2.2.1 Complete Algorithms/Exact Algorithms

Complete algorithms (or exact algorithms) guarantee to find optimum solutions to all finite-sized instances of the considered CO problem within bounded time. If no polynomial time algorithm to a CO problem is known (this is currently the case for $\mathcal{NP}$-hard problems), then the search for an optimum solution might need exponential running time in the worst case (Blum and Roli [2003]).

2.2.2 Approximate Algorithms/Heuristics

Approximate algorithms (or heuristics) forgo the guarantee to find optimum solutions in order to compute "good" solutions within a significantly reduced amount of time (Blum and Roli [2003]). In general, heuristics are distinguished into *constructive* and *local search algorithms*. Constructive algorithms are fast methods, which start from scratch, i.e. an empty partial solution, and continue to add components to the partial solution until the solution finally is complete. In contrast, a complete (nonempty initial) solution serves as input for local search algorithms. Afterwards the local search algorithm applies operators, which specify the way a solution can be modified, to the current solution; in this step the current solution is replaced by a better one. The determined set of operators is called *neighborhood structure*; the *neighborhood* of a solution $\mathcal{S}$ is the set of solutions deriving from having once applied an operator to $\mathcal{S}$. Constructive algorithms tend to produce significantly worse solutions than local search

algorithms. As a result, constructive and local search algorithms are typically combined in a single algorithm.

2.2.3 Metaheuristics

Local search algorithms that only replace solutions by better ones (the so-called *hill climbing* strategy) run the risk of converging to local optimum solutions, which do not have to be (global) optimum solutions. Conversely, *metaheuristics* combine basic heuristic methods in high-level frameworks to overcome non-optimum, local minimum solutions. In order to attain this objective, metaheuristics create a balance between *diversification*, i.e. the exploration of search space, and *intensification*, i.e. the exploitation of accumulated search experience (Blum and Roli [2003]).

This chapter is closed by presenting three different metaheuristics that are primarily included in the heuristic approach that is introduced in Chapter 6.

Simulated Annealing

Simulated Annealing (SA)(Černỳ [1985], Kirkpatrick et al. [1983]) is a strategy to escape from converging to local minimum solutions; a solution is allowed to be replaced by a worse solution. The probability of allowing moves to a worse solution is decreased (*cooling schedule*) within the search on a nature-inspired way (Blum and Roli [2003]). The general structure of SA is to initialize some starting temperature T at the beginning of the algorithm. A newly computed solution $\mathcal{S}'$ replaces the current solution $\mathcal{S}$ with probability according to the Boltzman distribution $e^{-\frac{c(\mathcal{S}')-c(\mathcal{S})}{T}}$, in this term $c(\mathcal{S})$ denotes the cost of a solution $\mathcal{S}$. The temperature T is reduced during the search; the choice of an appropriate cooling schedule is critical for the performance of the SA.

Large Neighborhood Search

Large Neighborhood Search (LNS) (Shaw [1997]) systematically explores neighborhoods in order to move from one promising area in the solution space to another one. *Removal* (a part of the solution is destroyed) and *insertion* (the destroyed solution is repaired) operators define the neighborhood of LNS. This metaheuristic is particularly suitable for instances having large neighborhoods or instances having neighborhoods containing only a few feasible solutions, which might result from restrictions like time windows (Blum and Roli [2003]).

Ropke and Pisinger [2006] propose an *adaptive LNS (ALNS)* that is inspired by the algorithms of Schrimpf et al. [2000], Shaw [1997]. Several destroy and fast repair operators are combined in one single step of the algorithm; operators for the successive modification of the solution are selected based on statis-

tics concerning their previous efficiency in the search, i.e. the search is *guided*. Instead of applying neighborhood structures in such a way that the current solution is only slightly modified, the algorithm of Pisinger and Ropke [2010] spends considerable computation time in one single move that is able to modify $30\% - 40\%$ of the current solution. Afterwards solutions are updated in a SA strategy. The heuristic approach that is presented in Chapter 6 also includes a SA strategy and some of the operators proposed by Ropke and Pisinger [2006]; however, the approach is classified as LNS because of a non-guided selection of operators.

Matheuristics

Nowadays, commercial MIP solvers can easily be embedded in users' implementations of common programming languages not only, but also commercial MIP solvers as well as machines have been highly improved and have become sufficiently faster. This is reason to the fact that the number of mathematical programming models that are used in heuristic frameworks rises in order to obtain high-quality solutions to complex problem definitions. The approach that combines mathematical models with heuristics is referred to as *matheuristic*. Archetti and Speranza [2014] surveys matheuristics for routing problems; matheuristics are classified into three different categories in this survey. *Decomposition approaches* divide problems into smaller sub-problems of which at least one is solved by using a MIP. *Improvement heuristics* use mathematical models to improve solutions that are found by heuristics. The third variant obtains heuristic solutions by *branch and price or column generation based approaches*. The matheuristic approach that is presented in Chapter 6 is a decomposition approach; a MIP is formulated to compute solutions to the sub-problem of assigning empty containers to cargo transportation requests. A further survey on matheuristics is given by Ball [2011]; a survey on matheuristics for rich vehicle routing problems is given by Doerner and Schmid [2010].

Chapter 3

Literature Overview

Problems arising in the field of moving containerized cargo and repositioning empty containers are a topical subject of high relevance from both, a scientific perspective as well as an economic and an ecological perspective. In this context especially the importance of operations in the hinterland of intermodal container terminals such as seaports or railway stations has to be emphasized. At container terminals containers are transshipped between different means of transportation for onward transportation. A bunch of surveys and reviews that has been recently published demonstrates the relevance and the actuality of tasks taking place at container terminals. Günther and Kim [2006] investigate container terminals and terminal operations, an overview of the transshipment of containers at a container terminal is given by Vis and De Koster [2003], Steenken et al. [2004] conduct a classification and a literature review of container terminal operations, a literature update of Operations Research (OR) at container terminals is presented by Stahlbock and Voß [2008], Macharis and Bontekoning [2004] review and discuss opportunities for OR in intermodal freight transport, an overview of container shipping and ports is provided by Notteboom [2004]. The following literature overview is restricted to scientific literature on applying quantitative methods to economic decision making in the field of drayage operations. Most of the papers consider one trucking company that has to transport cargo by trucks, whereby empty containers serve as transportation media for cargo. Empty containers have to be collected and repositioned. For this reason, a trucking company is also responsible to fulfill so-called unproductive movements.

Definition 3.1 (Unproductive Movement)**.** An **unproductive movement** is a part of a truck's route, in which the truck travels without having loaded any container or having loaded only empty containers.

In general, tractors, trailers and containers can be classified as *resources* (tractors, trailers, drivers, containers and so on). Smilowitz [2006] classify resources by their jobs; while containers and trailers are used for storage of cargo, but do not include a mechanism for locomotion (i.e. *non-autonomous* resources, Drexl [2012] that must be pulled by a compatible autonomous resource), tractors provide locomotion (i.e. *autonomous* resources, Drexl [2012]), but do not provide storage and require another resource (a driver) to operate. The subsequent literature overview either focuses on container or on trailer transportation. Depending on the transportation media, it is either assumed that tractors and drivers form one unit (trailer transportation), or that tractors, drivers and chassis form one unit (container transportation), i.e. tractors/trucks are always able to move in space on their own and no more than two types of resources need to be synchronized. In order to refer to container transportation, the term *truck* is used in the following. A truck is defined according to a *container vehicle* that is defined by Chung et al. [2007]. The definition distinguishes between three truck types:

Definition 3.2. A **truck** comprises a driver, a tractor and a chassis. Three types of chassis can be distinguished: 20-foot container chassis, 40-foot container chassis and combined chassis. Tractors having coupled the first or second chassis type are able to transport one 20-foot or 40-foot container respectively, while tractors having coupled the third chassis type are able to transport up to one 40-foot or two 20-foot containers. Depending on the coupled chassis, trucks are divided into **20-foot trucks**, **40-foot trucks** and **combined trucks**.

Sections 3.2 and 3.3 consider *full truckload (FTL)* problems, i.e. trailer transportation by tractor or 40-foot container transportation by 40-foot truck. The other types of trucks become interesting from Section 3.4; in this section a distinction is additionally made between different container sizes leading to *less-than truckload (LTL)* problems. In literature sources several terms are used to describe the movement of fully loaded containers, container handling operations ((de-)coupling and (un-)loading containers) and the repositioning of empty containers by truck. These terms are for example loads, container movements, transportation requests, tasks and so on. The term *hinterland request* is used in the following to refer to the different tasks a trucking company is confronted with.

Definition 3.3 (Hinterland Request)**.** The term **hinterland request** refers to one task or the combination of different tasks (empty container transportation, container handling and cargo transportation) of a trucking company to form one multi-task unit.

Container types are subdivided depending on container transportation taking place in pre- or end-haulage.

Definition 3.4. Export/outbound containers are fully loaded containers that have to be transported from a customer's (sender's) location to the terminal. An **Export** is a hinterland request referring to the associated export container transportation. **Import/inbound containers** are fully loaded containers that have to be transported from the terminal to a customer's (receiver's) location. An **import** is a hinterland request referring to the associated import container transportation.

Container movements may be further split into movements having previously known points of origin and destination (fully loaded container transportation) and movements, for which either the point of origin or the point of destination is known in advance and the missing location has to be examined during the solution making process (empty container transportation). This distinction is based on the fact that empty containers are assumed to be interchangeable; as soon as empty container movements are considered, the empty container transporting the cargo is not further specified. For instance, after a fully loaded import container has been transported from the terminal to the customer's location, where it has been unloaded, the obtained empty container can either be reused as an export container by transporting the container to a customer's location, where it will be filled again, or the container can be carried to a terminal/depot for stowage. We follow the notations of Smilowitz [2006] to distinguish between the different attributes of a container/truck movement:

Definition 3.5 (Well-defined and Flexible Tasks, Smilowitz [2006])**.** A **well-defined** task is given points of origin and destination in advance. A **flexible** task is given only one location (either the point of origin or the point of destination) beforehand. The point of origin or the point of destination of a flexible task is missing, and therefore is a matter of optimization. A hinterland request is called well-defined, if it is comprised of well-defined tasks only. A hinterland request is called flexible, if it starts/ends with a flexible task that has got an undetermined point of origin/destination.

The reminder of this chapter is structured as follows. Section 3.1 gives a classification on the different types of problems appearing in drayage. Sections 3.2 and 3.3 review transportation problems of 40-foot containers. Whereas Section 3.2 amplifies problems containing well-defined hinterland requests only, Section 3.3 expands to problems containing well-defined and flexible hinterland requests. Problems considering the simultaneous transportation of containers having different sizes, mainly 20-foot and 40-foot containers, are an-

alyzed in Section 3.4. Section 3.5 summarizes different challenges arising in container hinterland transportation.

3.1 Classification of Drayage Problems

The problems that are stated in Sections 3.2 and 3.3 consider the transportation of 40-foot containers/trailers only. These problems can be classified as FTPDP (Erera and Smilowitz [2008]) (see also Definition 2.29), in which trucks/tractors can carry a maximum of one good (40-foot container/trailer). Each customer defines two locations: one location where the good has to be picked up and another location where the good has to be delivered. The objective is to minimize costs like the total operating time or the total travel distance of the trucks. The trucks start and end their routes at the depot and visit all customers' locations, whereby the destination of a customer has to be visited immediately after its origin. If *time windows* are additionally given, then the problem is referred to as *Full Truckload Pickup and Delivery Problem with Time Window Constraints (FTPDPTW)*, in which trucks have to arrive at the different locations within the locations' time windows. Among others, Jula et al. [2005], Wang and Regan [2002] show how to transform the FTPDP/FTPDPTW into an asymmetric mTSP/mTSPTW (amTSP/amTSPTW) (as detailed in Section 3.2).

3.1.1 (Multiple) Traveling Salesman Problem

For a finite set of locations the TSP asks for a route, in which each location is visited exactly once by a salesman, thereby minimizes the total travel distance of the salesman. Problem definitions including time windows, in which the salesman has to arrive at the different locations, are called *Traveling Salesman Problem with Time Window Constraints (TSPTW)*. Karp [1972] proves the TSP to be $\mathcal{NP}$-hard. However, in practice, there are fast approximate algorithms as well as complete algorithms for solving large TSP instances. For instance, Helsgaun [2000] implemented the approximate algorithm of Lin and Kernighan [1973] so as to compute solutions of a superior level of solution quality to TSP instances containing up to 100,000 locations. One famous complete TSP algorithm is called *Concorde* (Applegate et al. [1995], Applegate et al. [2011]). Concorde is able to compute optimum solutions to instances like the 85,900-location instance of the *TSPLIB* (Reinelt [1991]). Nowadays, scientists are trying to optimally solve a world instance consisting of up to 1,904,711 locations (World-TSP-Pictures [2004]). However, for unknown reasons, optimum solutions to instances of such a magnitude can only be computed for the symmetric TSP. In practice, the asymmetric TSP (aTSP) seems to be much harder

to solve (Grötschel [2015]). An analysis of heuristics for the aTSP is presented by Johnson et al. [2007].

The mTSP that considers several salesmen states a generalization of the TSP. Bektas [2006] gives an overview of formulations and solution procedures for the mTSP. Although several complete mTSP algorithms are introduced in literature, one outcome of the overview of Bektas [2006] is that in contrast to the TSP, the largest optimally solved amTSP instances contain about 500 locations, while the largest optimally solved symmetric TSP instances contain about 100 locations (Gavish and Srikanth [1986]). The consideration of time windows at the different locations constitutes a further extension of the mTSP/amTSP that is called *(asymmetric) multiple Traveling Salesman Problem with Time Window Constraints (mTSPTW/amTSPTW)*. Computing solutions to reasonable-sized instances can prove to be very difficult for problem definitions including time window constraints (Williams [2013]). One of the reasons for this is that time window constraints are non-linear (see for example Constraints 5.17). However, linearizing time window constraints makes the problem hard to solve for general solution approaches, as the linearized constraints are not the facets of the polytope of the convex hull of the feasible solutions (Langevin et al. [1990], Wang and Regan [2002]). Surveys of time window constrained routing problems are for example given by Desroches et al. [1988], Solomon [1987], Solomon and Desrosiers [1988]. If the salesmen are given maximum capacities that have to be obtained, then the problem is called *Vehicle Routing Problem (VRP)*. Surveys of the VRP have been conducted by e.g. Laporte [1992], Toth and Vigo [2002].

3.1.2 Vehicle Routing Problems with Synchronization

The consideration of the simultaneous transportation of 40-foot and 20-foot containers transforms the FTL problem to a LTL problem. Trucks can transport either one 40-foot container or two 20-foot containers at the most. Compared to common definitions of routing problems, additional synchronization constraints are required for the construction of routes in this case. Synchronization constraints may lead to routes that are dependent on one another:

Definition 3.6 (Interdependence Problem, Drexl [2011, 2012])**.** In routing problems, the term **interdependence problem** refers to a change in one route r that may effect changes in routes different from r.

This special class of vehicle routing problems has become a highly topical subject of many literature sources in the recent years. A general definition is stated by Drexl [2011, 2012]:

Definition 3.7 (Drexl [2011, 2012])**.** The **Vehicle Routing Problem with Multiple Synchronization Constraints (VRPMS)** is a VRP, in which more than one vehicle may or must be used to perform a task.

Surveys of VRPMS are conducted by Drexl [2011, 2012]. Depending on the subject (i.e. task, operation, movement, load, resource) that needs to be synchronized, Drexl [2011, 2012] establishes a classification scheme for VRPMS. Drayage problems including the handling of different sized containers might ask for synchronization in the following aspects: trucks transporting two or more containers at the same point in time simultaneously perform more than one task, routes for trucks and flows for containers have to be synchronized, empty and fully loaded container transportation has to be synchronized whenever container (un-)loading operations are allowed to take place without the presence of trucks.

Multiple Traveling Salesman Problem with Time Windows and Precedences

The mICTP includes a sub-problem that is a VRPMS, which has been originally defined for an application different from container hinterland drayage. For the routing and scheduling of workers that have to perform a set of electricity maintenance jobs Goel and Meisel [2013] define the *Multiple Traveling Salesman Problem with Time Windows and Precedences (mTSPTWP)* that constitutes an extension of the mTSPTW. A mTSPTWP instance comprises a set $\mathcal{J}$ of multi-task jobs. Each job $j \in \mathcal{J}$ is represented by a task set $\mathcal{T}_j$. The different tasks that are combined in $\mathcal{T}_j$ have to be performed in a given sequence. Precedence constraints $\mathcal{P}_j \subset \mathcal{T}_j \times \mathcal{T}_j$ are introduced in order to ensure the compliance with the defined sequence of tasks $\mathcal{T}_j$ belonging to the same job j. A precedence constraint between two tasks $(\tau, \tau') \in \mathcal{P}_j$ specifies that task τ has been completed before task τ' is allowed to start.

3.2 Full Truckload Problem

Wang and Regan [2002] investigate a FTPDPTW, in which a homogeneous fleet of trucks (each having a capacity of one) has to serve a set of well-defined hinterland requests. A hinterland request defines a load to be moved from a pickup to a delivery location. Trucks have to arrive at the loads' pickup locations within a predefined time interval. Initially, trucks are scattered throughout the entire service region. The considered problem is an *open routing problem*; that means, no destination is specified at which the trucks have to arrive at the end of the *time horizon*. The service region contains one or more intermodal facilities. The objective is to serve as many hinterland requests as

possible and thereby minimize the total travel cost of the trucks' fleet, i.e. a solution remains feasible, if some of the hinterland requests are not executed. Figure 3.1 shows the routes of three trucks. A hinterland request i is depicted

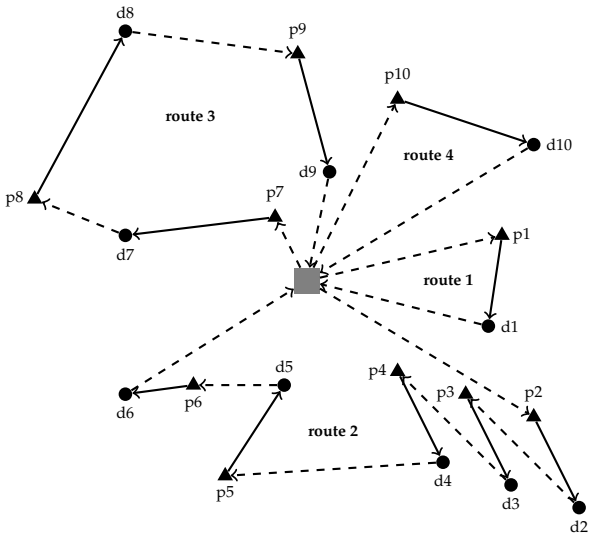


Figure 3.1: Representation of a full truckload problem due to Jula et al. [2005].

by the node pair (p_i, d_i) representing the transportation of one load from its pickup p_i to its delivery location d_i. As the problem is a FTL problem, the hinterland requests constitute sub-sets of the routes. This means that the loads' transportation (shown by solid lines) is already determined, while the order in which the hinterland requests are to be served and the assignment to the trucks (shown by dashed lines) remain a matter of optimization. Wang and Regan [2002] model the problem as an amTSPTW by introducing one node for each hinterland request. This means that the pairs (p_i, d_i) (solid lines) are merged into one node, with the node's point of origin differing from the node's point of destination. This implementation is the reason why an asymmetric problem is obtained: the costs of arcs entering a node are equal to distances to the pickup locations, while the costs of arcs leaving a node are equal to distances from the delivery locations. Nodes have assigned time intervals equal to the time windows of the pickup locations and *service times* (the minimum time a truck has to stay at a node), which are equal to the sum of service times obtained at pickup and delivery locations with the travel duration from pickup to delivery locations. For the computation of solutions to the formulation, Wang and Regan [2002] propose an iterative solution approach named *window partition based (WPB) method*. The WPB method replaces time window constraints with binary variables. Each iteration consists of solving two different models, both

representing a simplification of the same problem definition. The first model is a copy of the original amTSPTW formulation in which arcs (l_1, l_2) are excluded whenever time constraints between the end points of the time intervals of l_1 and l_2 are not satisfied (regardless of whether the movement (l_1, l_2) is feasible or infeasible). This model is called *over-constrained*, since feasible solutions to this model are also feasible solutions to the original problem definition. The second model is a copy of the original amTSPTW formulation in which infeasible arcs (l_1, l_2) are added whenever time constraints between the start point of the time interval of l_1 and the end point of the time interval of l_2 are satisfied. This model is called *under-constrained*, since feasible solutions to this model are not necessarily feasible solutions to the original problem definition. The objective value of the over-constrained/under-constrained model yields an upper/lower bound on the objective value of the original problem definition. That is, the smaller the gap between the objective values of the two models, the better the approximation to the objective value to the original problem formulation. Smaller time window widths of the nodes result in smaller gaps between the models. As a result, Wang and Regan [2002] partition the time windows into sub-sets and introduce a node for each time window sub-set. A load is served, when a truck visits one of the nodes representing a sub-set of the load's time window. As the computation time increases for larger problem sizes, the challenge of the approach is to determine a width for the time window partition (i.e. the number of nodes) that achieves a good solution quality within an adequate computation time. Therefore, the WPB method consists of several iterations. In each iteration the preceding partition of time windows is repartitioned until a time window width is achieved that leads to high-quality solutions.

The main difference between the problem definition of Wang and Regan [2002] and the problem definition of Jula et al. [2005] is that the entire set of hinterland requests must be served in the problem definition of Jula et al. [2005]. Consequently, the objective minimizes the total distance of unproductive movements (i.e. the dashed lines in Figure 3.1) since the movement of loads is predefined and therefore cannot be modified by any solution making process. The authors build an amTSPTW and add *social constraints* in order to ensure that the time horizon is not exceeded by the duration of a truck route. Jula et al. [2005] additionally propose a two-phase exact algorithm based on dynamic programming. In the first phase, a set of feasible solutions is generated. In order to cover all hinterland requests at minimum costs, the second phase solves a Set Partitioning Problem (SPP) (see also Definition 2.32), which contains these truck routes that have been computed in the first phase. Since the computation time increases with the time window widths, optimum solutions

can only be computed to instances containing up to 13 hinterland requests in the case of wide time windows. Therefore, Jula et al. [2005] propose two heuristics. The first heuristic approach differs from the exact approach by solving the SPP by a heuristic genetic algorithm. The second heuristic approach is an insertion heuristic method (Jaw et al. [1986]) that iteratively adds hinterland requests to an initial empty set of routes. In more detail, one iteration comprises two steps. In the first step the unserved hinterland request causing the least cost when inserting it into the currently computed set of routes is determined. In the second step this hinterland request is inserted into the cheapest position. A new route is initialized whenever a hinterland request cannot be added to the current set of routes.

3.3 Repositioning of Empty Containers

If the FTL problem includes additionally the repositioning of empty containers or trailers, then the most of the literature sources groups different tasks in one multi-task unit. Two types of hinterland requests can be distinguished depending on whether tasks accrue in pre- or end-haulage. Each type comprises three tasks.

Definition 3.8 (Outbound Full Hinterland Request, Zhang et al. [2010])**.** The term **outbound full (OF) hinterland request** refers to tasks taking place in pre-haulage. The order of the tasks is given as follows:

1. Transportation of an empty container to the sender's location (flexible task, missing origin)

2. Loading the empty container at the sender's location (container handling operation)

3. Export of the fully loaded container from the sender's location to the terminal (well-defined task)

Definition 3.9 (Inbound Full Hinterland Request, Zhang et al. [2010])**.** The term **inbound full (IF) hinterland request** refers to tasks taking place in end-haulage. The order of the tasks is given as follows:

1. Import of a fully loaded container from the terminal to the receiver's location (well-defined task)

2. Unloading the fully loaded container at the receiver's location (container handling operation)

3. Transportation of the empty container from the receiver's location (flexible task, missing destination)

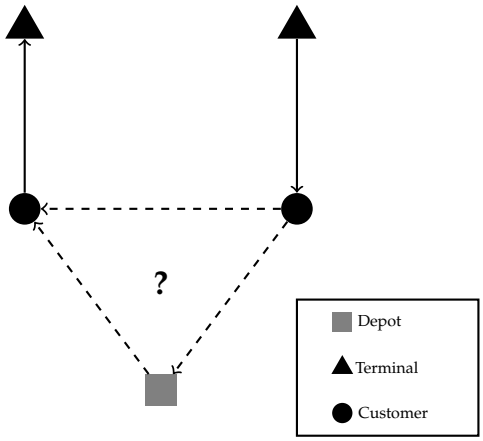


Figure 3.2: Different possibilities for combining routes.

The introduction of flexible tasks adds a new optimization component to the problem definition, since the missing locations are to be determined by the solution making approach. As a result, these approaches often consist of at least two phases: one phase assigns empty containers or trailers to flexible tasks and thereby specifies the missing location, i.e. flexible tasks are grouped to form well-defined tasks. The other phase constructs routes from the well-defined tasks. An example is shown in Figure 3.2. An OF request is depicted on the left and an IF request is depicted on the right. Fully loaded container transportation between terminal and customer or vice versa is represented by solid lines. In this example, empty containers can be obtained from/delivered to the depot or between the customers (receiver and sender) shown by the dashed lines.

Definition 3.10 (Street-turn, Jula et al. [2006])**.** A **street-turn** is the direct movement of an empty container from a receiver to a sender or vice versa.

The second task of OF and IF hinterland requests is a container handling operation that is allowed to take place either with (Section 3.3.1) or without the presence of a truck (Section 3.3.2)[1].

[1](Un-)loading containers without the presence of a truck is a typical proceeding that is e.g. applied in the port of Rotterdam (Veenstra [2005]). After decoupling the container at the customers' location, the truck has the opportunity to pick empty containers up that have been delivered the day before. Further advantages of (de-)coupling containers and trucks at customers' locations can be seen from Sterzik et al. [2015].

3.3.1 No Separation

Ileri et al. [2006] optimize drayage operations taking place in intermodal transportation between truck and rail on the same day. The research focuses on the minimization of the total cost of the operating times of a heterogeneous fleet of tractors; the tractors are given individual operating costs. Furthermore, tractors are given a point of origin and destination, and a minimum and a maximum bound on the operating time. A set of hinterland requests defines trailers to be collected and delivered to specific locations within specific time windows. Some of the hinterland requests are well-defined whereas others demand/supply empty trailers and, therefore, are flexible. The problem is given an unlimited availability of empty trailers. However, the pickup and the delivery of empty trailers takes additional time and distance. Ileri et al. [2006] propose a *Column Generation (CG)* methodology in which tractors that share the same attributes are grouped together. The CG comprises several cycles. Each cycle optimally solves a SPP *relaxation* that includes only a sub-set of all feasible routes. *Dual values* are used in order to identify columns (routes) having *negative reduced costs*. The columns are separately chosen for the different tractor groups; whenever the maximum number of routes for one tractor group has been generated, the algorithm proceeds creating routes for the next tractor group. Finally, the new routes are added to the SPP formulation and the next cycle is initiated. The circulation process terminates, when no more columns that have negative reduced costs can be identified. Afterwards, *branch and bound* techniques are used to solve the integral formulation of the SPP. The proposed algorithm is able to compute solutions to instances containing around 100 hinterland requests within three minutes. Since the obtained solutions must not be the optimum solutions to the entire problem formulation, Ileri et al. [2006] compare solutions to small-sized instances with solutions that are computed by another exact approach. The other approach enumerates all of the columns beforehand and solves the IP formulation afterwards. The enumeration uses *search trees* (one for each tractor group), a *feasibility matrix* and *dominated routes*. The exact approach is able to solve instances including a small number of hinterland requests (5 to 28); both algorithms compute the same solutions to the considered benchmark set.

In the problem formulation of Imai et al. [2007] a set of flexible OF and IF hinterland requests defines the movement of empty and fully loaded containers among senders, receivers and one intermodal terminal by truck. Trucks and containers are not allowed to separate:

Definition 3.11 (Stay-with, Macharis and Bontekoning [2004])**.** A trailer/container is (un-)loaded in a **stay-with procedure**, when the operation is accompanied by the tractor/truck.

As container handling operations are carried out in a stay-with procedure, a truck has to visit at least three different locations (customer, terminal and the container's origin/destination) for the entire transportation of one specific container. The set of trucks comprises two different homogeneous sub-sets containing either *owned* or *chartered* trucks. While the maximum time for operating is shorter for owned trucks, the travel costs of chartered trucks are conversely more expensive. The problem's objective is to minimize the total travel distance of the trucks and thereby serve all of the hinterland requests. Imai et al. [2007] propose a solution method based on Lagrangian Relaxation, in which the subgradient method is used to compute the Lagrangian multipliers. The solution method firstly assigns empty containers to flexible IF and OF requests by solving an AP in polynomial running time (refer to Remark 2.15). Afterwards, routes are constructed from the obtained well-defined requests by an implementation of the $\mathcal{NP}$-hard *Generalized Assignment Problem (GAP)*. In order to compute high-quality *Lagrangian multipliers*, the AP and the GAP have to be solved several times; this step is especially time consuming for the GAP. For this reason, Imai et al. [2007] replace the GAP by a formulation of the $\mathcal{NP}$-hard *Bin Packing Problem (BPP)*, which yields an upper bound for the GAP. In contrast to the GAP, the BPP can be efficiently solved by heuristics (like the *first fit decreasing algorithm*) in short computation time. On the whole, however, the solution methodology of Imai et al. [2007] profits from a problem definition, in which trucks are restricted to start and end their routes at one terminal and no additional time windows at the locations are given.

Caris and Janssens [2009] extend the problem definition of Imai et al. [2007] by introducing time windows, in which the trucks have to arrive at the different locations; the authors propose a FTPDPTW formulation of the obtained problem definition. In order to compute lower bounds on the solution quality, the feasibility of routes is relaxed by underestimating the total number of trucks that are needed to serve the entire set of hinterland requests (Currie and Salhi [2003]). Caris and Janssens [2009] construct an initial solution by a two-phase insertion heuristic approach. In a first phase, OF and IF requests get assigned the missing points of origin and destination. The objective in this phase is an intermediate between a saving function, which considers waiting and travel times that are obtained between two locations, and a cost function, which penalizes not selecting a specific assignment. In a second phase, a heuristic insertion method inserts the well-defined hinterland requests that

have been computed in the first phase into the cheapest (regarding waiting times) feasible positions of the truck routes. Afterwards, a local search heuristic algorithm is invoked; three different local search operators (changing of empty container assignments, combining of two routes, removing and reinserting of one request) are applied to improve the initial solution. This heuristic algorithm of Caris and Janssens [2009] is improved by Caris and Janssens [2010] two-fold. On the one hand, the constructive algorithm is replaced by a multistart approach. This means that the first phase is invoked multiple times; each time the costs of the assignments of empty containers to OF and IF requests are adapted. At the end, the best overall solution is improved by the local search algorithm. On the other hand, the three local search operators of the local search algorithm of Caris and Janssens [2009] are integrated in a *deterministic annealing* framework. Deterministic annealing (also known as *threshold accepting*, Dueck and Scheuer [1990]) is a local search algorithm that replaces a solution by a neighborhood solution whenever the difference between the objectives of the two solutions becomes smaller than a certain threshold.

As can be seen from Section 1.2.3, some regions are confronted with a lack or an overrun of empty containers resulting from imbalances in trade. As a result, Zhang et al. [2009] distinguish between two more types of hinterland requests in addition to OF and IF requests. Each of the newly introduced hinterland requests is flexible and comprises exactly one task. In order to overcome a surplus of empty containers, empty containers are transported to terminals for onward transportation:

Definition 3.12 (Outbound Empty Hinterland Request, Zhang et al. [2010])**.** An **outbound empty (OE) hinterland request** specifies an empty container to be delivered to a specific terminal (missing origin).

In order to balance occurrences of empty containers in regions that are confronted with a lack of empty containers, empty containers arrive at terminals for collection:

Definition 3.13 (Inbound Empty Hinterland Request, Zhang et al. [2010])**.** An **inbound empty (IE) hinterland request** specifies an empty container to be collected at a specific terminal (missing destination).

Depending on whether an export- or an import-oriented area is considered, the problem definition includes either IE or OE hinterland requests only. Therefore, the problem definition of Zhang et al. [2009] includes up to three different types of hinterland requests, which define the transportation of containers among several depots, customers' locations and one terminal. The entire set of hinterland requests has to be served by a homogeneous fleet of trucks. Each

truck starts its route from one specific depot and has to end its route at any of the depots within a given time horizon; the points of origin and destination need not necessarily be the same. Furthermore, trucks have to arrive at the different locations within specific time windows; containers are (un-)loaded in a stay-with procedure. Each depot is a huge stowage of empty containers, i.e. it is able to store every empty inbound container and it is able to provide every empty outbound container. The problem's objective is the minimization of the total distance of unproductive movements, which can be prevented by fulfilling street-turns instead of making detours via depots for empty container stowage. Zhang et al. [2009] formulate the problem as an amTSPTW with multiple depots. In the underlying graph, hinterland requests are represented by nodes and container storage operations at the depot are completely reflected by arcs. This can be done since there is only way to perform a street-turn, namely an outbound request has to be visited immediately after an inbound request has been visited. Therefore, all other combinations of two requests require that a container storage operation at the depot that causes the least detour has to be performed in between. This means that arcs either represent travel operations between the locations of its incident nodes or, in the case of not connecting inbound and outbound requests, arcs represent the detour via the depot together with the appropriate container storage operation. The authors compare two different heuristics computing solutions to problems of real-world size. The first heuristic approach is a *greedy cluster method* that removes infeasible arcs from a constructed solution and reassigns cheapest feasible arcs to the separated nodes afterwards. The second heuristic approach is a *Reactive Tabu Search (RTS)*, which randomly exchanges parts of the solution that has been constructed by the greedy cluster method. Battiti and Tecchiolli [1994] extend the well-known *Tabu Search (TS)* algorithm (Gendreau [2003], Glover [1986]) to the RTS. In a nutshell, TS is a metaheuristic approach that uses the history of the search to escape from non-optimum, local minimum solutions. A *tabu list* keeps track of the most recently visited solutions and forbids moves toward them. One step of TS is to move to the best neighborhood solution that is not located in the tabu list and to store the corresponding neighborhood solution in the tabu list afterwards. The *tabu tenure* denotes the length of the tabu list, i.e. if more neighborhood solutions are located in the tabu list than allowed by the tabu tenure, some solutions will be deleted from the tabu list (mostly in a *First In First Out (FIFO) technique*). If the tabu tenure is allowed to vary during the search, then the approach is called RTS. The RTS that is proposed by Zhang et al. [2009] modifies the tabu tenure in accordance with the length of the time period, in which solutions have been previously visited. Zhang et al. [2011] apply a similar RTS to a problem definition that differs from the previ-

ous problem by including only one depot that has stored a limited number of empty containers. The complexity noticeably increases for problems including a limited number of empty containers.

Zhang et al. [2010] extend the problem of Zhang et al. [2009] to the *Inland Container Transportation Problem (ICT)*. The ICT includes more than one terminal and the objective is to minimize the total operating time of the trucks. Zhang et al. [2010] propose a solution methodology for the ICT, which is a modification of the WPB method of Wang and Regan [2002]. In order to build over- and under-constrained models, only one time window partitioning width is estimated. As a result, the two models are solved exactly once, i.e. in contrast to the original WPB method, the algorithm of Zhang et al. [2010] is not iterative. In the computational study, the modified WPB approach is compared with the RTS method of Zhang et al. [2009]; while the objective values are nearly the same, the computation time of the modified WPB approach is much shorter.

Nossack and Pesch [2013] propose two exact mathematical formulations of the ICT: an amTSPTW formulation (Zhang et al. [2010]) and a FTPDPTW formulation. The FTPDPTW formulation is based on an alternating graph representation (Asbach et al. [2009]), i.e. a bipartite graph in which one side of the partition only comprises pickup nodes, while the other side only comprises delivery nodes. Pickup nodes refer to flexible inbound requests (missing destination), and delivery nodes refer to flexible outbound requests (missing origin). Furthermore, nodes representing depots, which serve as sources and sinks for empty containers, are included in both sides of the partition. This is due to the fact that container storage operations taking place at a depot might lead to routes visiting the same location (depot) several times. In order to allow several visits of the same location within one route, Nossack and Pesch [2013] duplicate nodes representing the depot several times and add the duplicates to both sides of the partition. The number of depot duplicates depends on the number of trucks, depots and hinterland requests of the problem's instance. Nossack and Pesch [2013] present a 2-stage heuristic approach to compute solutions to real-world sized ICT instances: the first stage constructs routes, the second the stage improves routes. The route construction comprises four phases. The first and second phase minimize the total travel time and the total waiting time of trucks. The first phase optimally solves an AP, whereby the alternating graph representation is used as instance for the AP, i.e. pickup nodes are assigned to delivery nodes. The well-defined requests that have been paired by the AP are assigned to feasible sequences regarding time windows in the second phase. In the third phase, an AP is formulated for the assignment of starting depots to the sequences, i.e. trucks represented by the depots are assigned to routes represented by the sequences. The sequences get assigned these depots as an

end that are located the nearest to the sequence's final location. The starting times and the ending times of the sequences are computed in the fourth phase; this is done by solving either an exact LP formulation or a heuristic approach that successively assigns starting times from the sequences' sinks to sources. In the second stage, an *ejection chain heuristic approach* (Pesch and Glover [1997]) improves the constructed solution by applying multiple local search operators (*relocating a single node, exchanging tails or depots of the routes*) in one single iteration. In contrast to the approach of Zhang et al. [2010], the algorithm of Nossack and Pesch [2013] is independent from the time windows' width in terms of solution quality as well as computation time.

A further study on the ICT is provided by Sterzik and Kopfer [2013]. Sterzik and Kopfer [2013] formulate a MIP on the basis of a graph representation that comprises two types (for trucks and containers) of binary three-index arc decision variables. The decision variables state whether or not a truck/container traverses an arc. The MIP includes synchronization constraints, which connect the different types of decision variables. In order to compute solutions to instances of real-world size, a modified version of the well-known *savings algorithm* (Clarke and Wright [1964]) constructs an initial solution. The initial solution is improved by a TS algorithm combining three different local search operators (relocating of requests, exchanging of requests, inserting short routes into other routes). The TS applies several times relocate and exchange operators in order to intensify the search; a restart of the tabu list is combined with a diversification strategy of Taillard [1993].

Wang et al. [2016] investigate a variation of the ICT considering neither time window constraints nor limitations on the number of trucks. The aim of the study is to compute high-quality solutions to very large instances consisting of around 300 hinterland requests (in general ICT instances comprise around 75 hinterland requests, for more details on the size of instances refer to Section 7.1). Wang et al. [2016] introduce an iterative matheuristic approach, which repeats two steps until a certain stop criterion is met. In the first step, the set of hinterland requests is partitioned into sub-sets. The savings algorithm is used to identify sub-sets that have a great potential for improvement. A tabu list prevents returns to recently considered sub-sets. In the second step, a MIP is solved for each of the sub-sets. The MIP is inspired by the research of Sterzik and Kopfer [2013] and constructs routes for trucks.

3.3.2 Separation

Francis et al. [2007], Smilowitz [2006] investigate a problem definition in which a set of tractors has to carry loaded and empty trailers among senders, re-

ceivers, equipment yards and intermodal yards. The considered multi-task types are similar to OF and IF requests. Tractor routes start and end at one central depot. The problem definition additionally includes time windows in which the tractors have to arrive at the different locations. In order to finish their routes, the tractors are given a time horizon in which they have to arrive at the depot. The objective is multicriterial; the number of routes and the total duration time of routes is minimized. In contrast to scientific works, which are mentioned in Section 3.3.1, it is allowed to perform the second task (i.e. the (un-)loading operation) of OF and IF requests without the presence of a tractor:

Definition 3.14 (Drop-and-pick, Macharis and Bontekoning [2004])**.** A trailer/-container is (un-)loaded in a **drop-and-pick procedure**, when the tractor/truck decouples the trailer/container at the customer's location. During loading and/or unloading the tractor/truck is able to carry out other activities.

Smilowitz [2006] builds an *asymmetric Vehicle Routing Problem (asymmetric VRP)* model that is inspired by the formulation of Bodin et al. [2000]. The depot, each well-defined task and each possible execution of a flexible task introduces a node to the underlying graph representation. This means that a node might represent two locations as well as the subsequent movement between the locations (similar to the formulation presented in Section 3.2). Furthermore, not all of the nodes that represent possible executions of flexible tasks must be visited, instead, it has to be ensured that exactly one possible execution of each flexible task is visited. In addition, Smilowitz [2006] proposes a two-phase solution approach. First, a set of feasible solutions is constructed by combining branch and bound, CG and *Linear Relaxation (LR)* techniques. Second, a SPP that includes the routes of all feasible solutions that have been constructed in the first step is optimally solved. New routes are generated by solving the $\mathcal{NP}$-hard version of the *Shortest Path Problem (SP)*, which includes time window constrains and social constraints. Francis et al. [2007] improve the solution methodology of Smilowitz [2006] in two ways. On the one hand, Francis et al. [2007] introduce a methodology that defines node-specific radii for feasible executions. These radii limit the number of the feasible executions of flexible tasks to only contain these nodes that are no more than the specific radius apart. On the other hand, Francis et al. [2007] combine the approach of Smilowitz [2006] with a heuristic *greedy randomized procedure*, which generates a richer set of routes.

Reinhardt et al. [2012] present an exact approach, which computes solutions to a $\mathcal{NP}$-hard problem definition that includes several terminals, time window restrictions and a set of flexible OF and IF requests. Furthermore, containers are allowed to be (un-)loaded in either a stay-with or a drop-and-pick proce-

dure. The exact approach is a column enumeration approach, which covers the requests by paths, whereby a path refers to either a pair comprising an IF and an OF request, or the separate service of an IF/OF request. The efficiency of the exact approach is evaluated by a computational study that involves instances that are based on real-world data. Due to the time window restrictions of the instances, the number of feasible routes is small. Consequently, the complete enumeration of all feasible paths is impossible. Reinhardt et al. [2012] propose different models and objectives, like the minimization of costs obtained by paths, the minimization of the number of used trucks and the balance of the number of empty containers, which are stored at the terminals.

Sterzik et al. [2012] extend the ICT in two different aspects, namely *drop-and-pick procedure* and *container sharing*. In order to implement container handling operations taking place in a drop-and-pick procedure, Sterzik et al. [2012] introduce two different time windows at the same location of a customer; a truck has to decouple the container within the first time window, and either the same or another truck has to couple the container within the second time window after the container has been (un-)loaded. For the purpose of investigating container sharing, Sterzik et al. [2012] add several cooperating trucking companies to the original problem definition of the ICT. Each trucking company operates its own hinterland requests and depots. Depots have assigned a homogeneous fleet of trucks and a sufficiently large set of empty containers. The consideration of more than one trucking company adds a further synchronization component to the problem formulation. Sterzik et al. [2012] adapt the mathematical model and the heuristic solution procedure of Sterzik and Kopfer [2013][2] to solve instances of the newly obtained problem definition. In the computational study, Sterzik et al. [2012] compare two different scenarios. The first scenario prohibits trucking companies to share empty containers among each other, while the second scenario permits empty containers to be interchanged among several trucking companies. Sterzik et al. [2012] recognize a huge cost saving potential arising from container sharing in particular for tight time windows at the different locations. However, putting a successful container sharing cooperation into practice is difficult, due to the fear of individual trucking companies to be not compensated themselves, while the coalition in total above-average benefit from the agreement. This disadvantage is also reflected by the approach of Sterzik et al. [2012] that does not include the possibility of a fair mechanism for sharing empty containers among the different companies.

Braekers et al. [2013] investigate a problem definition differing in three aspects from the ICT. First, the problem definition includes the opportunity to perform container handling operations in a drop-and-pick procedure. Second,

[2]The article has earlier been available as a working paper, Sterzik and Kopfer [2012].

the problem definition includes a single depot, which only serves as the point of start and end of truck routes, while empty containers are stored at container terminals instead of the depot. Third, the objective is a lexicographic minimization of the number of used trucks and the total travel distance of the trucks. Braekers et al. [2013] formulate the problem as an amTSPTW and compare two different heuristic solution strategies for the computation of solutions to large instances: a sequential and an integrated approach. The sequential approach consists of two steps. The first step solves a TP, which assigns empty containers to flexible hinterland requests (due to Remark 2.20, it is possible to optimally solve the TP in polynomial time, e.g. with the algorithm of Ford and Fulkerson [1956]). The second step solves a FTPDPTW, which constructs truck routes from the well-defined requests that have been computed in the first step. Once more the techniques of Jula et al. [2005], Wang and Regan [2002] are applied in order to transform the FTPDPTW formulation to an amTSPTW formulation, i.e. introducing one node, which represents the entire transportation of a container between two different locations. The integrated approach simultaneously solves the combined problem of empty container assignment and route construction; Braekers et al. [2013] suggest two different formulations of the entire problem. Each of the two formulations is harder to solve than the sequential approach. Similar to the formulation of Smilowitz [2006], in the first formulation one node is introduced for each feasible empty container allocation (for more details see also Braekers et al. [2010]). The disadvantage of this formulation is the large number of nodes in the network whenever real-world sized instances are considered. Therefore, in the second formulation arcs reflect an intermediate stop (like container storage operations at the terminals) taking place between certain types of nodes (like two outbound/inbound hinterland requests). This proceeding results in an amTSPTW formulation, which is inspired by the models of Ileri et al. [2006], Zhang et al. [2009]. In both, the second formulation of the integrated approach as well as the second step of the sequential approach, an amTSPTW has to be solved. However, exactly solving the amTSPTW for real-world sized problem instances is very hard (see e.g. Jula et al. [2005]). Therefore, Braekers et al. [2013] propose a deterministic annealing approach that is based on the algorithms of Bräysy et al. [2008], Caris and Janssens [2010] to obtain heuristic solutions to the amTSPTW. All in all, five local search operators are combined in the deterministic annealing approach: three local search operators (*relocate, 2-opt*, exchange*) try to minimize the total travel distance of the trucks and two local search operators (reinserting all nodes of respectively one or several (randomly selected or shortest) routes into other routes) try to minimize the number of used trucks. The initial solution of the deterministic algorithm is computed by an insertion heuristic

method (Jula et al. [2005], see also Section 3.2). A single-phase algorithm is obtained that simultaneously minimizes the number and the total travel distance of trucks. However, since simultaneous approaches tend to neglect the minimization of the number of trucks (Bent and Van Hentenryck [2006], Homberger and Gehring [2005]), Braekers et al. [2013] introduce a two-phase algorithm in which both phases use the same deterministic annealing approach, but, the first phase minimizes the number of trucks, while the second phase minimizes the total travel distance of the trucks. Lower bounds on the objective values of both approaches are obtained by the WPB method (Wang and Regan [2002], see also Section 3.2). The study of Braekers et al. [2013] examines that the integrated approach clearly outperforms the sequential approach.

The problem definition of Xue et al. [2014] differs from the problem definition of Braekers et al. [2013] in two aspects. First, the locations are not given time windows. Trucks only have to finish their routes within a specific time horizon. Second, while the objective also defines the minimization of the number of used trucks and the total travel distance of the trucks, in contrast to the previous problem definition, it is not lexicographic. Xue et al. [2014] present lower bounds on both terms of the objective. The problem is modeled as a variant of the amTSP. In order to allow multiple visits of the same customer's location within a truck route caused by container handling operations taking place in a drop-and-pick procedure, the underlying graph definition includes two nodes, which both represent the same customer's location. Similar to the previous approach and the graph representation of Zhang et al. [2009], container stowage operations at the depot are reflected by arcs connecting certain nodes. Xue et al. [2014] propose a metaheuristic for the computation of solutions. The constructive method invokes an insertion heuristic approach (Solomon [1987]). Afterwards, the initial solution is improved by a TS approach, which includes three local search operators (*relocate, exchange, reinserting of all nodes of one route into other routes*). After each step the observation of precedence times between nodes representing the same customer's location is checked.

The FTPDPTW proposed by Meisel and Kopfer [2014] is a transportation problem of containerized cargo within the hinterland of seaports not only, but also arises in several other applications. Therefore, a general distinction is made between autonomous and non-autonomous resources (refer to the beginning of this chapter), which are needed to serve the set of requests. Each autonomous resource is able to carry one non-autonomous resource at the most. Requests are given a pickup and a delivery location. The locations define time windows for the visit and the completion of services. An empty non-autonomous resource has to be moved by an autonomous resource to the pickup location. At the pickup location the non-autonomous resource is filled

with cargo. Afterwards, the autonomous resource carries the loaded non-autonomous resource to the delivery location where the cargo is unloaded. Each (un-)loading operation is allowed to be performed in both procedures (stay-with or drop-and-pick). Therefore, it is part of the decision making process to assign procedures to (un-)loading operations. Compatibility restrictions between non-autonomous resources and requests, as well as between autonomous resources and non-autonomous resources are given. Autonomous resources start and end their routes at one depot within a given time horizon. Non-autonomous resources start and end their routes at individual not necessarily equal locations. The objective minimizes three criteria: the weighted sum of the total travel distance, the total duration of all routes, and the number of unserved requests. By duplicating pickup and delivery nodes that might be visited twice by the same autonomous resource, Meisel and Kopfer [2014] formulate the considered problem as a generalization of the mTSPTW. Since the problem is $\mathcal{NP}$-hard, (optimum or small gap) solutions to real-world sized instances cannot be computed by commercial MIP solvers. This fact holds true, even when strengthening the mathematical formulation. As a result, Meisel and Kopfer [2014] propose a LNS (see also Section 2.2.3). The constructive algorithm generates an initial solution in three steps. First, each non-autonomous resource gets assigned an empty route. Second, the requests are sorted by increasing end times. Third, an insertion heuristic method assigns requests one after another to these non-autonomous resources into whose routes the current request can be feasibly inserted at the least cost. A solution of the third step is feasible, if it is possible to generate routes for autonomous resources from it. For this purpose, so-called carriages are constructed out of the assignment of requests to non-autonomous resources. A carriage comprises the smallest connected sub-tour of a non-autonomous resource. In other words, each carriage is a triple, which defines the pickup location, the delivery location and the non-autonomous resource. An iterative process assigns the most urgent carriages to autonomous resources, or decides that a request is not served by any of the autonomous resources. Afterwards, a local search algorithm and a LNS are alternately invoked in order to improve the constructed solution. After each modification, the feasibility of the solution is checked using the same method that is also applied in the third step of the constructive algorithm. The local search algorithm is comprised of three local search operators for non-autonomous resources (*relocate, insert, exchange*) and one local search operator (*relocate*) for autonomous resources. The LNS consists of three different destroy operators (*worst, random, obstacle*) and the same insertion heuristic method that is also invoked in the third step of the constructive algorithm. Whereas the first two destroy operators are an adaption from Ropke and Pisinger [2006]

(see also Section 6.3.2 for more details), the third destroy operator is newly developed and aims in including unserved requests into the routes of a solution. Meisel and Kopfer [2014] add two different mechanisms to the LNS that select destroy and repair operators with respect to their performance within previous iterations. In this way the LNS becomes adaptive(Ropke and Pisinger [2006]).

3.4 Multi-size Container Transportation

In recent years, more and more scientific articles investigate the combined transportation of containers having different sizes, namely 20-foot and 40-foot containers. Since 20-foot and 40-foot containers are used most widely almost all over the world (Zhang et al. [2015]), it is hardly surprising that this topic has been only recently investigated. In Europe (except Finland and Sweden) and Asia, road vehicles are restricted to transport 20-foot and 40-foot containers (Popović et al. [2012], Vidović et al. [2011], see also Section 1.1 for more details about the usage of ISO containers). The introduction of two different commodities (containers) turns the original problem formulation from a FTL to a LTL problem; combined trucks have a capacity of two TEU. As can be seen from the reminder of this section, the introduction of two different commodities considerably increases the problem's complexity. The problem definition often has to be simplified in order to find an optimum solution.

Chung et al. [2007] are among the first authors considering a heterogeneous set of containers. In their work, Chung et al. [2007] propose variations of mathematical models for the TSP and the VRP as well as heuristics for several drayage problems, which have to be served by Korean trucking industries. The authors define a basic FTPDP: a homogeneous fleet of trucks has to serve a set of well defined requests within a given time horizon. Each request describes the transportation of a container from a pickup location to a delivery location. The objective is to minimize the number of trucks, which are needed to serve the entire set of requests. Chung et al. [2007] build an amTSP model of the problem by introducing one node for each container's pickup and delivery pair (refer to Section 3.2). The basic problem definition is extended in several ways, like implementing time window constraints. Finally, a new mathematical model is build for the introduction of 20-foot and 40-foot containers, which have to be transported by a heterogeneous fleet comprising 20-foot, 40-foot and combined trucks (refer to Definition 3.2). This new model is a variant of the VRP in which container's pickup and delivery pairs are implemented by two different nodes that are connected by precedence constraints. The objective is two-fold: the minimization of the total travel distance of trucks or the minimization of the total weighted travel distance in which the set of trucks is

divided into company-owned and mandated trucks. Even though the amTSP and the VRP are both $\mathcal{NP}$-hard problems, there are fast solution methods for the amTSP in practice (see also Section 3.1.1). Therefore, Chung et al. [2007] propose a heuristic approach that only computes solutions to the problem definition that includes multiple commodities. The proposed heuristic approach is based on an insertion heuristic approach of Rosenkrantz et al. [1977] in which requests are randomly selected and tried to be inserted into the best feasible insertion position of all currently computed truck routes.

In the $\mathcal{NP}$-hard *Vehicle Routing Problem with backhauls (VRPB)*(Mingozzi et al. [1999], Toth and Vigo [1997]) that is investigated by Vidović et al. [2011] an unlimited homogeneous fleet of combined trucks has to transport empty and fully loaded 20-foot and 40-foot containers. The trucks start and end their routes at one terminal. A hinterland request is either a well-defined import or a well-defined export (refer to Definition 3.4). The maximum number of requests that are served by one route of a truck is four, because of the problem definition prohibiting the separation of trucks and containers. Vidović et al. [2011] build a multiple matching integer programming model in which the different routing possibilities are enumerated by binary decision variables. This means that each feasible truck route (sequence of one to four requests) introduces a decision variable; the number of the indices of the variable equals the number of requests in the sequence, i.e. it ranges from one to four. In order to solve large-sized problem instances, Vidović et al. [2011] propose a heuristic approach. Based on a continually updated calculation, which estimates the effect of merging requests into one route, the requests are greedily assigned to routes.

The work of Vidović et al. [2012] extends the problem definition of Vidović et al. [2011] by limiting the size of the truck fleet to a finite number and by introducing time windows in which trucks have to reach the different locations. Afterwards, Vidović et al. [2012] compare the performances of two different MIP formulations for the newly obtained problem definition. The first MIP model is an extension of the mathematical model of Vidović et al. [2011], while the second MIP model is a formulation that is inspired by the VRP with simultaneous pickups and deliveries (Mingyong and Erbao [2010]). The implementation of the formulation of Vidović et al. [2011] obtains very promising results in the computational studies of Vidović et al. [2012].

Popović et al. [2012] develop a metaheuristic approach for the problem definition of Vidović et al. [2012]; the initial solution is constructed by a modification of the classical *sweep method* (Gillett and Miller [1974]) and improved by a *variable neighborhood search* heuristic approach, which includes two local search operators (*relocate, exchange*), after.

Lai [2013], Lai et al. [2013] investigate the routing of a heterogeneous truck fleet with single and double container loads, i.e. a finite number of combined trucks and 20-foot trucks has to serve OF and IF hinterland requests defining the transportation and the handling of 20-foot containers. In the problem definition one port provides containers for OF hinterland requests and stores containers obtained by IF hinterland requests. While the fulfillment of street-turns between IF and OF hinterland requests is permitted, the separation of trucks and containers is prohibited (only stay-with procedure is allowed). Time windows are not given, but IF hinterland requests have to be served before OF hinterland requests within a truck route. Due to the routes' order of IF and OF requests, this problem is also a VRPB in which routes include four requests the most. The objective is two-fold: one the one hand, handling costs at the port have to be minimized, i.e. street-turns are preferred, and, on the other hand, routing costs have to be minimized, whereby a truck that transports two containers is more expensive than a truck that transports one container. Lai [2013], Lai et al. [2013] introduce a mathematical model that contains synchronization constraints. Afterwards, a metaheuristic approach is proposed in which two different criteria are used for evaluation. One criteria minimizes the actual objective and the other criteria minimizes the total travel distance. The metaheuristic approach comprises a constructive algorithm and a local search. The constructive algorithm constructs a feasible solution by a variant of the savings algorithm. The subsequent local search tries to improve the initial solution by combining two different local search operators (*relocate, exchange*).

Lai [2013] extends the problem definition of Lai et al. [2013] by considering a homogeneous fleet of trucks, which have a capacity of C containers ($C \in \mathbb{N}$). Since customers asks for a large number of containers, the problem definition allows the split of loads. The obtained problem definition is a split delivery VRPB in which routes comprise at most $2C$ requests (combining C IF hinterland requests with C OF hinterland requests). In contrast to the objective of Lai et al. [2013], the objective of Lai [2013] minimizes the total travel distance of the trucks. Lai [2013] propose a mathematical model, a heuristic approach and a metaheuristic approach. The heuristic approach consists of two phases; a TS algorithm constructs routes consisting of either OF or IF requests (*split VRP* phase), afterwards the former constructed routes are merged by seven different rules that are implemented in heuristics (*merging phase*). Finally, the best solution regarding the objective is selected. The metaheuristic approach sequentially repeats three phases until some stop criterion is met: a *split VRP phase*, a *merging phase* and an *adaptive guidance phase*. In contrast to the heuristic algorithm, the merging phase is solved by an implementation of an IP formulation. The adaptive guidance phase analyzes the current solution and detects

the areas of improvement. Afterwards, a new iteration is invoked in which the split VRP phase receives adjusted input parameters.

As already mentioned in Section 1.2.3, the imbalances of empty containers are caused by global and regional levels. Literature sources commonly investigate container imbalances appearing in the entire hinterland region on global level (Braekers et al. [2013], Imai et al. [2007], Zhang et al. [2010]). Conversely, Schönberger et al. [2013] consider a regional problem in which customers' locations are connected with the imbalances of empty containers. This different consideration results in hinterland requests defining the transportation of containers only, i.e. container transportation requests and container handling operations are no longer combined in one request. The set of hinterland requests contains two types of transportation requests. On the one hand, well-defined transportation requests define the movement of fully loaded 20-foot/40-foot containers between customers' locations and one terminal or vice versa. On the other hand, flexible transportation requests define the provision/collection of empty 20-foot/40-foot containers at/from customers' locations. A homogeneous fleet of combined trucks has to depart from and return to a single depot within a given time horizon. Furthermore, the depot is a sufficiently large storage of empty containers. There are once again two possibilities to replace the missing location of flexible requests: depot or street-turn. The objective minimizes the total travel distance of trucks. Schönberger et al. [2013] extend a MIP formulation of a LTL PDP model that implements the recursive arrival time update for short cycle prevention in vehicle flows, which is presented by Sigurd et al. [2000]. The model of Schönberger et al. [2013] combines three different sub-problems: the assignment of missing locations to flexible requests, the construction of container flows and the generation of truck routes. The computational studies of Schönberger et al. [2013] establish the high complexity of the considered problem definition. Often, the solver failed even to identify a feasible solution.

Nordsieck et al. [2016, 2017] adapt the problem definition of Schönberger et al. [2013] in two points. First, time windows are added to locations. Second, the objective minimizes the total operating time of trucks. A heuristic algorithm is proposed that sequentially repeats three phases until some stop criterion is met. The first phase solves an AP and thereby assigns missing locations to flexible requests. The objective of the AP minimizes two criteria: the total waiting time and the total travel distance of trucks. Since the problem decomposes for 20-foot and 40-foot containers (see also Chapter 6.1.3), it is possible to separately solve two different AP's without influencing the solution quality. After this phase, the remaining problem is a *PDP with Time Window Constraints (PDPTW)*. In the second phase, an insertion heuristic approach constructs a

sufficiently large number of feasible routes. These routes are the input for a *Set Covering Problem (SCP)*, which minimizes the total operating time of trucks subject to covering all requests by routes. The solution is improved by a local search that includes three different local search operators (*rearrange, shift, exchange*). The third phase adapts the costs of assignments, which are needed by the objective function of the first phase. The two problems for 20-foot and 40-foot containers are considered independently of one another in this step.

Zhang et al. [2015] investigate the cost of one trucking company that is located in an export-oriented area. The trucking company has to serve a set of IE, OF and IF hinterland requests for 20-foot and 40-foot containers. Each request defines the handling and the transportation of empty and fully loaded containers among one terminal, one depot and several customers' locations. The locations do not have a defined time window. A finite, homogeneous fleet of combined trucks has to start and end its routes at the depot within a specific time horizon. Moreover, the depot is a sufficiently large storage of empty containers. The problem definition prohibits the separation of truck and container ((de-)coupling operations are performed in a stay-with procedure). Zhang et al. [2015] introduce the *full-twin assumption*, which is a new constraint that restricts trucks carrying two 20-foot containers to finish both containers' requests before starting a new request. The objective minimizes the total travel time of trucks. Zhang et al. [2015] build a MIP that distinguishes the two states of a truck: different locations and different filling levels. The MIP becomes an extension of the amTSP with social constraints through the definition of the transition between the different states. However, since the MIP involves sequence dependent distances between locations in the objective and in some constraints, even small-sized instances cannot be solved by commercial solvers. Therefore, Zhang et al. [2015] propose a representation of the solution as a series of numbers; three different tree search strategies are applied to this representation: one depth-first complete search, one search that cuts similarities and one parallel search that starts with a certain drayage request. In order to compute solutions to real-world sized instances, the RTS of Zhang et al. [2009] is adapted to the multi commodity case. A pair-making algorithm generates an initial solution of the RTS algorithm.

Caballini et al. [2015] investigate the transportation of 20-foot containers. The study does not only include IF (*import trips*) and OF (*export trips*) hinterland requests, but also so-called *inland trips* that define the movement of empty/fully loaded containers between specific inland locations. Caballini et al. [2015] minimize the unproductive movements of a set of combined trucks by maximizing the trucks' utilization. The considered problem definition assumes that the routes are round trips, i.e. empty containers have to be returned to

their origin at the end of a route. However, it is possible to reduce the travel distance by performing a street-turn, in which trips have to be performed in a given sequence (import – inland – export). Various types of time windows are introduced for the trips, the terminals, the companies and the depots. Moreover, vessel departure times and the EU regulation of driving hours are considered. Caballini et al. [2015] propose a matching MIP, which is based on the work of Vidović et al. [2012]. This means that the two types of binary decision variables are introduced: variables using three indices represent routes comprising three trips and variables using one index represent routes comprising individual trips. The values of the variables are set to one whenever the corresponding route is included in the solution; the values are set to zero otherwise. The objective is to minimize costs, which are comprised of the costs of combining trips (considering costs for reusing containers, delay), the repositioning costs of containers and the costs of individually serving trips.

In contrast to the problem definition of Caballini et al. [2015], the problem definition of Daham et al. [2016] considers a heterogeneous truck fleet comprising 20-foot trucks and combined trucks. The trucks are given soft and hard time horizons, as well as weight limitations for the transportation of 20-foot containers. The trips define container movements among several customers' locations and one port. Furthermore, a trip defines a container to be transported from an origin to either one or two nominated customers' locations within specified time windows. The full-twin assumption (Zhang et al. [2015]) has to hold true for trips that are given two customers' locations. Some of the containers have specified a point of destination, while other containers require the assignment of a point of destination out of a set of possible destinations. The costs are comprised of the total travel duration and the total over time of the trucks' soft time horizons. Daham et al. [2016] formulate a MIP based on the formulation of Vidović et al. [2012] for a problem definition containing import trips only. Afterwards, the MIP is extended to also include the combination of import, inland and export trips.

Great emphasis is put on the minimization of the number of simplifications that are made to the solution space, when defining the mICTP. The reason for this is that methodical approaches should work closely to conditions given by real-world applications. The mICTP is firstly defined by Funke and Kopfer [2015]. Afterwards, Funke and Kopfer [2016][3] extend the definition to also consider time windows at the different locations. A detailed definition of the mICTP is given in Chapter 4. An exact approach (Funke and Kopfer [2016]) for the mICTP is shown in Chapter 5. Chapter 6 presents a heuristic approach that arises from the neighborhood search shown by Funke and Kopfer [2015]. Com-

[3]The article has been available in 2014 as a working paper Funke and Kopfer [2014].

putation results are presented in Chapter 7. In addition, Chapters 4 to 8 discuss similarities and differences between the definition and the chosen methods for the mICTP and for the former mentioned problems that are presented by literature sources.

3.5 Challenges

Table 3.1 gives an overview of the different problem definitions that are discussed in this chapter. All but one paper (Vidović et al. [2011]) considers a fleet consisting of a finite number of trucks.

Source	TW	flexible tasks	separation	LTL	serve all requests	heterogeneous autonom. fleet
Wang and Regan [2002]	yes	no	no	no	no	no
Jula et al. [2005]	yes	no	no	no	yes	no
Ileri et al. [2006]	yes	yes	no	no	yes	yes
Imai et al. [2007]	no	yes	no	no	yes	yes
Caris and Janssens [2009, 2010]	yes	yes	no	no	yes	yes
Zhang et al. [2009, 2010, 2011] Nossack and Pesch [2013] Sterzik and Kopfer [2013]	yes	yes	no	no	yes	no
Wang et al. [2016]	no	yes	no	no	yes	no
Smilowitz [2006] Francis et al. [2007]	yes	yes	yes	no	yes	no
Reinhardt et al. [2012]	yes	yes	yes	no	yes	no
Sterzik et al. [2012]	yes	yes	yes	no	yes	yes
Braekers et al. [2013]	yes	yes	yes	no	yes	no
Xue et al. [2014]	no	yes	yes	no	yes	no
Meisel and Kopfer [2014]	yes	yes	yes	no	no	no
Chung et al. [2007]	yes	no	no	yes	yes	yes
Vidović et al. [2011]	no	no	no	yes	yes	no
Vidović et al. [2012] Popović et al. [2012]	yes	no	no	yes	yes	no
Lai [2013], Lai et al. [2013]	no	yes	no	yes	yes	yes
Schönberger et al. [2013]	no	yes	yes	yes	yes	no
Nordsieck et al. [2016, 2017]	yes	yes	yes	yes	yes	no
Zhang et al. [2015]	no	yes	no	yes	yes	no
Caballini et al. [2015]	yes	yes	no	yes	yes	no
Daham et al. [2016]	yes	yes	no	yes	yes	yes
Funke and Kopfer [2015]	no	yes	yes	yes	yes	no
Funke and Kopfer [2016]	yes	yes	yes	yes	yes	no

Table 3.1: Summary of problems defined by literature sources.

As can be seen from the table, the difficulty to formulate operations taking place in the sector of drayage is reflected by a bunch of problem definitions, whose complexity increases. Moreover, several different objectives are considered by literature sources. Whereas the research of Wang and Regan [2002] minimizes unproductive movements, while maximizing the number of served

hinterland requests, Braekers et al. [2013], Vidović et al. [2011] minimize the number of used trucks and the total travel distance. Other scientific sources, like Nossack and Pesch [2013], Sterzik and Kopfer [2013], Sterzik et al. [2012], Zhang et al. [2010], minimize the total operating time of trucks. The costs that are composed of the total travel distance and the over times of the trucks' time horizons are investigated by Daham et al. [2016]. The objective that is investigated by Lai [2013], Lai et al. [2013] combines the minimization of truck dependent travel costs with the minimization of port handling costs. Meisel and Kopfer [2014] minimize the weighted sum of the total travel distance, the total finish time of all truck routes and the number of unserved requests. The minimization of the costs of routes, the number of used trucks and the balance of empty containers that are stored at terminals is considered by Reinhardt et al. [2012]. Daham et al. [2016] minimize the total travel cost, the costs of reusing containers and the lateness costs.

If the analysis additionally includes further real-world constraints, then problem definition becomes even more complex. Namboothiri and Erera [2008] consider a problem definition that is inspired by U.S. port terminals. U.S. port terminals have mostly implemented appointment systems. Therefore, Namboothiri and Erera [2008] add an appointment-based access control system to a problem definition in which one drayage company serves import and export requests. The drayage company is based at a single port and operates a homogeneous fleet of trucks. The extension by an appointment-based access control system means that the time horizon is partitioned into distinct time intervals; each time interval allows only a given amount of trucks to arrive at the port. Similar to the problem definition of Wang and Regan [2002], not all of the requests have to be served. Hence, the objective minimizes the number of used trucks, on the one hand, and maximizes the number of unserved requests, on the other hand. Namboothiri and Erera [2008] define a FTPDP model that includes social constraints; moreover, the authors develop a heuristic approach that is based on CG.

Fur the purpose of implementing dynamic container transportation, Zhang et al. [2014] extend the mathematical formulation and the graph representation of Zhang et al. [2009] to a mixed-zero-one nonlinear programming model. Afterwards, the authors show different strategies, like the WPB-method (Wang and Regan [2002]), for the handling of interruptions.

In order to formulate further restrictions that are given by real-world problems, Coslovich et al. [2006] introduce various constraint sets. The problem definition includes a time horizon that comprises several days. The most of the hinterland requests and their time windows are known to the trucking company during the day. Since the trucking company is only able to estimate

the demand of the next day, the problem becomes stochastic as well as dynamic. Each day decisions must be taken on partially unknown information (Crainic and Laporte [1997], Coslovich et al. [2006]). As a result, the proposed solution approach dynamically solves the problem day by day. The objective function combines several criteria. On the one hand, routing costs that are usually proportional to the time/length of a route have to be minimized. On the other hand, the driver-tractor assignment states a conflicting cost driver; while drivers wish to finish their routes closely to their domiciles, the most of the times it is even more important that tractors finish their routes closely to the terminals of the next-days' requests (Taylor and Meinert [2000], Taylor et al. [2001]). Finally, the repositioning costs of empty containers have to be considered; because of the heterogeneous set of containers, street-turns between specific hinterland requests are prohibited. Coslovich et al. [2006] propose an IP formulation and create a heuristic LR solution approach from three subproblems of the IP formulation.

Cheung et al. [2008] investigate the impact of regulatory policies in Hong Kong port in cross-border drayage productivity. Two policies, which significantly restrict the efficiency of drayage operations, are considered. The *4-up-4-down* policy forbids driver, tractor, chassis and container to separate, i.e. the drop-and-pick procedure is prohibited. The *1-driver-1-tractor* policy determines that tractors are operated by specific drivers.

Chapter 4

The Multi-size Inland Container Transportation Problem

This chapter defines the mICTP, i.e. the drayage problem that is the key element of the investigation of this thesis. In order to obtain an abstract representation by mathematical formulas of problems arising in practical applications, there is a need for a precise mathematical definition of input, output and constraints. The mICTP is defined in such a way that only a few simplifications are made to the solution space leading to methodical approaches working very close to conditions given by real-world applications. As shown in Table 3.1, the mICTP integrates all but one of the headings (the heterogeneous trucks' fleet) that are analyzed in more detail in the previous literature overview. Furthermore, the mICTP (Funke and Kopfer [2014], Funke et al. [2016]) is an extension of the ICT (Zhang et al. [2010]). The simultaneous consideration of 20-foot and 40-foot containers as well as the possibility to (de-)couple containers in a stay-with or a drop-and-pick procedure (refer to Definitions 3.11 and 3.14) are added to a widely studied definition (e.g. Nossack and Pesch [2013], Sterzik and Kopfer [2013], Zhang et al. [2010]) of a container transportation problem that takes place in the hinterland region of a seaport.

4.1 Problem Definition

The mICTP considers one trucking company that is responsible for the transportation of containerized cargo among several customers (senders and receivers), terminals and one depot. Three different types of the tasks of the

trucking company can be distinguished; the trucking company primarily serves transportation requests for containerized cargo. Because of the trucking company additionally providing empty containers for the transportation of cargo, the trucking company has to transport and reposition empty containers. The third type of task is stated by container handling operations like (un-)loading containers with cargo. All tasks are known in advance. The objective is to minimize the total travel distance of the trucking company's homogeneous fleet of combined trucks (see Definition 3.2). For the transportation, cargo is grouped into 20-foot and 40-foot containers, whereby the size of the container is specified by the corresponding transportation request. The depot is a sufficiently large repository for empty containers; this stowage is capable of handling all supply and demand for empty containers specified by the problem's instance. Empty containers that share the same size are interchangeable. The problem formulation combines two sub-problems: a routing problem for trucks and an assignment problem for empty containers to cargo transportation requests. Empty containers can be assigned to cargo transportation requests in one of two different ways (refer to Figure 3.2). First, it is always possible to obtain as well as store empty containers in the depot. Second, empty containers that have become available by inbound requests can be directly moved to outbound requests (*street-turn* (Jula et al. [2006]), Definition 3.10). The trucking company is responsible to serve the entire set of cargo transportation requests including all of its sub-tasks (empty container provision and container handling operation). The different tasks have to be assigned to trucks and sequenced on truck routes resulting in all tasks being served while meeting several restrictions like time windows at the different locations and a time horizon limiting the duration of a route. Trucks have to start and end their routes at the depot.

4.1.1 Hinterland Requests

The different tasks (empty container transportation, container handling and cargo transportation) of a trucking company are grouped in one hinterland request (refer to Definition 3.3). Zhang et al. [2010] distinguish between four types of hinterland requests: OF/IF requests accrue in pre-/end-haulage (Definitions 3.8 and 3.9) and OE/IE requests improve the balance of empty containers in import-/export-oriented areas (Definitions 3.12 and 3.13). These definitions are adapted to the mICTP: the different sizes of containers have to be included in the definitions of the requests and the possibility to serve container (un-)loading operations in a drop-and-pick procedure has to be added; the latter might require additional time for (de-)coupling containers from trucks.

Definition 4.1. An **OF hinterland request** defines the order of three tasks:

1. Transportation of an empty container of a specific size (20- or 40-foot) to the sender's location (flexible task, missing origin)

2. Decision whether the container and the truck separate (drop-and-pick procedure) or not (stay-with procedure)

   Decoupling of the container from the truck (only for drop-and-pick procedure)

   Loading of the empty container at the sender's location (container handling operation)

   Coupling of the container by the same or another truck (only for drop-and-pick procedure)

3. Export of the fully loaded container from the sender's location to the terminal (well-defined task), *Decoupling of the container from the truck*

An **IF hinterland request** defines the order of three tasks:

1. *Coupling of a fully loaded container by a truck*, Import of the fully loaded container from the terminal to the receiver's location (well-defined task)

2. Decision whether the container and the truck separate (drop-and-pick procedure) or not (stay-with procedure)

   Decoupling of the container from the truck (only for drop-and-pick procedure)

   Unloading of the fully loaded container at the receiver's location (container handling operation)

   Coupling of the container by the same or another truck (only for drop-and-pick procedure)

3. Transportation of the empty container from the receiver's location (flexible task, missing destination)

An **OE hinterland request** comprises one task:

1. Transportation of an empty container of a specific size (20- or 40-foot) to the terminal (flexible task, missing origin), *Decoupling of the container from the truck*

An **IE hinterland request** comprises one task:

1. *Coupling of an empty container by a truck*, Transportation of the empty container from the terminal (flexible task, missing destination)

Apart from the construction of truck routes and the assignment of empty containers to cargo transportation requests, a decision has to be made, whether

the second task of OF and IF requests is served in a stay-with or in a drop-and-pick procedure. If the container is (un-)loaded in a drop-and-pick procedure, then the container decouples from the truck; either the same or another truck collects (and couples) the container after it has been (un-)loaded. Coupling and decoupling operations need additional time. The mICTP assumes that each customer has the opportunity to separate trucks and containers. Customers and terminals are given container pickup and delivery times, i.e. the starting time of the first task and the third task of OF and IF requests and the single task of OE and IE requests has to lie within a particular interval (*hard time window*). Furthermore, it is assumed that the second task of an OF/IF request is immediately served after the truck has been arrived at the customer's location. If the second task of an OF/IF request is served in a stay-with procedure, the accompanying truck is allowed to leave the customer's location after the container handling operation has been completed. If the second task of an OF/IF request is served in a drop-and-pick procedure, then the truck that picks the container up is permitted to arrive at the customer's location at any point in time after the container handling operation has been completed, i.e. the customer's time interval must not be observed for the collection of the container.

4.1.2 Instance

The instance comprises primarily three sets: a homogeneous fleet of combined trucks denoted by $\mathcal{T}$, a set of containers denoted by $\mathcal{C}$, and an ordered set of hinterland requests denoted by $\mathcal{R}$. For a container $c \in \mathcal{C}$ let teu_c state the size in TEU of c. The set of hinterland requests $\mathcal{R}$ can further be subdivided:

Sub-set	Description	Location s_i	Location e_i
OF	OF hinterland requests	Sender	Terminal
IF	IF hinterland requests	Terminal	Receiver
OE	OE hinterland requests	Terminal	Terminal
IE	IE hinterland requests	Terminal	Terminal
OF_{40} / OF_{20}	OF 40-/20-foot hinterland requests	Sender	Terminal
IF_{40} / IF_{20}	IF 40-/20-foot hinterland requests	Terminal	Receiver
OE_{40} / OE_{20}	OE 40-/20-foot hinterland requests	Terminal	Terminal
IE_{40} / IE_{20}	IE 40-/20-foot hinterland requests	Terminal	Terminal

Table 4.1: Different sub-sets of hinterland requests.

Each request $i \in \mathcal{R}$ is given two locations s_i and e_i together with two time windows $[\mathrm{start}_{s_i}, \mathrm{end}_{s_i}]$ and $[\mathrm{start}_{e_i}, \mathrm{end}_{e_i}]$ in which trucks have to arrive at the corresponding location for the first time. For two locations s_i, e_i the function

$\mathrm{dist}(s_i, e_i)$ states the time/distance needed by a truck to move from s_i to e_i; the function dist is restricted to observe the triangle inequality. For a request $i \in \mathrm{OF}$, s_i is equal to the sender's location and e_i to the terminal, which are specified by i. For a request $i \in \mathrm{IF}$, s_i is equal to the terminal and e_i to the receiver's location, which are specified by i. For a request $i \in \mathrm{IE} \cup \mathrm{OE}$, s_i and e_i are both equal to the terminal that is specified by i. For a full hinterland request $i \in \mathrm{IF} \cup \mathrm{OF}$, the duration of (un-)loading a container is denoted by load_i. The durations of (de-)coupling containers $\mathrm{cpl}\,/\,\mathrm{dcpl}$ are assumed to be the same at each location. Finally, the instance specifies a single depot that is denoted by d together with a time horizon $[0, H]$ in which trucks have to leave and enter d.

4.2 Solution

A graph representation of the routes of a truck fleet $\mathcal{T}$ that comprises three trucks (blue, black and yellow) is depicted by Figure 4.1. All in all, the truck-

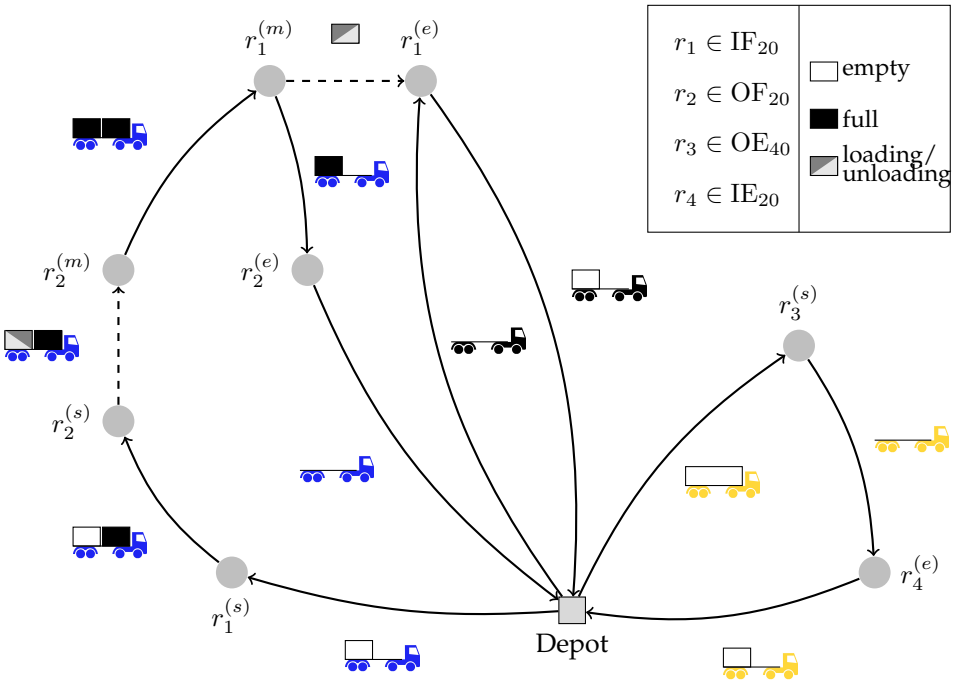


Figure 4.1: Example of a solution to an instance of the mICTP.

ing company has to serve four hinterland requests $r_1 \in \mathrm{IF}_{20}, r_2 \in \mathrm{OF}_{20}, r_3 \in \mathrm{OE}_{40}, r_4 \in \mathrm{IE}_{20}$. The empty hinterland requests r_3, r_4 introduce one node $r_3^{(s)}, r_4^{(e)}$ representing the corresponding terminal, whereas the full hinterland

requests r_1, r_2 introduce three nodes. The composition of nodes representing tasks belonging to requests r_1, r_2 is as follows: the nodes $r_1^{(s)}, r_2^{(e)}$ represent the requests' terminals, while the customers' locations are duplicated, i.e. the pairs $(r_1^{(m)}, r_1^{(e)})$ and $(r_2^{(s)}, r_2^{(m)})$ represent the same location of a customer. This duplication has been made in order to depict (un-)loading operations that are served in a drop-and-pick procedure. That is, dotted arcs between node pairs of the same customer's location represent (un-)loading operations. If a dotted arc is traversed by a truck, then the container is (un-)loaded in a stay-with procedure; if the dotted arc is not traversed by a truck, then the corresponding container is (un-)loaded in a drop-and-pick procedure. Furthermore, the depot that is the point of origin and the destination of the trucks' routes is depicted by the square.

The blue truck couples an empty 20-foot container at the depot and moves from the depot to the terminal of the first hinterland request ($r_1^{(s)}$). Here, the truck couples a fully loaded 20-foot container. The truck then carries the two containers to the sender's location of the second hinterland request ($r_2^{(s)}$), where the empty container is loaded in a stay-with procedure ($(r_2^{(s)}, r_2^{(m)})$). Afterwards, the truck carries the two containers to the receiver's location of the first hinterland request ($r_1^{(m)}$). In order to unload the fully loaded 20-foot container that has been collected at the terminal of the first hinterland request ($r_1^{(s)}$) in a drop-and-pick procedure, the truck decouples the container at the receiver's location ($r_1^{(m)}$). The truck travels to the terminal of the second hinterland request ($r_2^{(e)}$) and decouples the remaining 20-foot container. Finally, the truck finishes its route at the depot.

The black truck travels from the depot to the receiver's location of the first hinterland request ($r_1^{(e)}$). Here, the truck collects the 20-foot container that was unloaded in a drop-and-pick procedure after it was delivered to the receiver's location ($r_1^{(m)}$) by the blue truck. At the end, the truck carries the empty container to the depot, where the container is stored and the truck finishes its route.

The yellow truck carries an empty 40-foot container from the depot to the terminal of the third hinterland request ($r_3^{(s)}$). Afterwards, the truck moves to the terminal of the fourth hinterland request ($r_4^{(e)}$), where the truck collects an empty 20-foot container. The truck carries the empty container to the depot for stowage and finishes its route.

Chapter 5

Exact Approach

As can be seen from Chapter 3 some standard concepts for formulating a mathematical model and an underlying graph representation of drayage problems have been elaborated in literature sources over the past years. Among others Braekers et al. [2013], Jula et al. [2005], Wang and Regan [2002], Zhang et al. [2010] introduce one node that represents an entire FTL transportation request; this technique results in a formulation of the problem as an asymmetric routing problem. Nossack and Pesch [2013], Xue et al. [2014] duplicate nodes to allow several visits of the same location within one truck route. Arcs reflect storage operations in the formulations of Braekers et al. [2013], Xue et al. [2014], Zhang et al. [2009], if it has been determined that a storage operation at the depot takes place between the visits of certain nodes. Caballini et al. [2015], Daham et al. [2016], Ileri et al. [2006], Reinhardt et al. [2012], Smilowitz [2006], Vidović et al. [2011, 2012] show CG and matching models to cover the routes of trucks. Often (Braekers et al. [2013], Imai et al. [2007], Nordsieck et al. [2016, 2017], Nossack and Pesch [2013]) the two sub-problems of assigning empty containers to flexible tasks (AP/TP) and routing trucks (amTSP(TW)/FTPDP(TW)) are distinguished in order to take standard forms of network models. In this chapter a mathematical model for the formulation of the mICTP is shown. Some of the former mentioned proven methods from literature sources are combined with newly developed techniques. The content of this chapter is based on the work of Funke and Kopfer [2014, 2016]. Section 5.1 defines the underlying graph of the mathematical model. The two sub-problems of assigning containers (Section 5.2) and constructing truck routes (Section 5.3) are separately formulated at first. In a second step, the two models are coupled (Section 5.4) in order to obtain one model covering the entire problem. Section 5.5 shows two implementations of different objectives, namely the minimization of the total travel distance and the minimization of the total operating time of trucks.

5.1 Graph Definition

The MIP formulation follows the definition of the instances of the mICTP introduced in Section 4.1.2. An underlying weighted directed graph $G = (V, A)$ is used to build the mathematical model. The chosen graph is inspired two-fold. On the one hand, the representation of tasks by nodes and arcs is similar to the graph definition of Zhang et al. [2009]. On the other hand, the chosen graph definition assumes that nodes are allowed to be visited exactly once. Since storage operations may require several visits of the depot, nodes representing the depot are duplicated as also proposed by Nossack and Pesch [2013].

The node set V of the graph consists of the representatives of tasks together with two locations that need not necessarily be different (refer to Jula et al. [2005], Wang and Regan [2002]). The two locations are the point of start and end of the represented task. A task is either a storage operation that takes place at the depot or a sub-task of a hinterland request (one of the three tasks of an OF/IF request or the unique task of an OE/IE request; refer to Definition 4.1). Both types of tasks might be combined with a (de-)coupling operation. As a result, a node $v \in V$ is given a location pair $(\mathrm{orig}_v, \mathrm{dest}_v)$ where the represented task starts and ends, a time window $[\mathrm{start}_v, \mathrm{end}_v]$ in which a truck has to visit the node and a service time (serv_v) of the task during which the truck has to stay at the node. Unless otherwise stated, if the pair $(\mathrm{orig}_v, \mathrm{dest}_v)$ denotes the same location $l = \mathrm{orig}_v = \mathrm{dest}_v$, then the time window $[\mathrm{start}_v, \mathrm{end}_v]$ is equal to the time window $[\mathrm{start}_l, \mathrm{end}_l]$ that is defined by l.

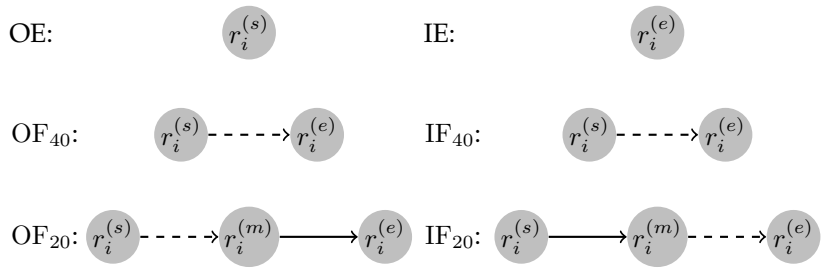


Figure 5.1: Graph of the different request types (Funke and Kopfer [2016]).

An arc $(v, w) \in A$ represents tasks that have to be served between nodes v and w. The majority of arcs represents the movement from the ending location dest_v of the arc's tail to the starting location orig_w of the arc's head, which might be combined with a (de-)coupling operation that has to be executed between v and w. A further possibility is that arcs are the representatives of (un-)loading operations. If an (un-)loading operation is performed in a drop-and-pick procedure (see also Definition 3.14), then the corresponding customer's location is

visited twice by either the same or different trucks. In order to enable several visits of the same location that are required by the drop-and-pick procedure, customers' locations are duplicated. This means that an arc $(v, w) \in A$ represents an (un-)loading operation, if v and w are the representatives of the same OF/IF hinterland request with dest_v and orig_w being equal to the location of the customer. In order to highlight the different arc types in the figures in this chapter, arcs representing moving operations are depicted by consistent lines, while arcs representing (un-)loading operations are depicted by dashed lines.

5.1.1 Hinterland Requests

The graph representation of a hinterland request $i \in \mathcal{R}$ is shown in Figure 5.1. In the case $i \in \mathrm{OE} \cup \mathrm{IE}$ is an empty hinterland request, it is sufficient to let the entire request i be represented by one node v, whereby $v = r_i^{(e)}$ for $i \in \mathrm{IE}$ and $v = r_i^{(s)}$ for $i \in \mathrm{OE}$. This means that this node represents a decouple ($i \in \mathrm{OE}$) or couple ($i \in \mathrm{IE}$) operation that takes place at the terminal specified by i in the time window of the terminal. The representation for full container transportation $i \in \mathrm{OF} \cup \mathrm{IF}$ is more complex. Hinterland requests defining the movement and the handling of 20-foot and 40-foot containers are treated differently.

Two nodes represent the movement and the handling of 40-foot containers, i.e. $i \in \mathrm{OF}_{40} \cup \mathrm{IF}_{40}$. The idea for the representation of fully loaded 40-foot container transportation steams from Jula et al. [2005], Wang and Regan [2002] (see also Section 3.2), who introduce a single node v to model the direct movement from the container's origin $\mathrm{orig}_v = s_i$ to the container's destination $\mathrm{dest}_v = e_i$, whereby $v = r_i^{(e)}$ for $i \in \mathrm{OF}_{40}$ and $v = r_i^{(s)}$ for $i \in \mathrm{IF}_{40}$. This representation is possible since the routes of fully loaded 40-foot containers are well-defined FTL transportation tasks, and therefore, predetermined. Since node v represents two locations, which are given different time windows, a single time window of v that reflects both time windows has to be formulated in a next step. The underlying assumption of the drop-and-pick procedure, which restricts the first visit of a customer's location to lie within a specified time window, while the second visit of a customer's location is allowed to exceed this time window, leads to a different formulation of the time windows for nodes representing OF and IF requests. If $i \in \mathrm{OF}_{40}$ denotes an outbound request, then the second visit of a customer is considered by node $r_i^{(e)}$ that represents the fully loaded container transportation. Therefore, the time window of $r_i^{(e)}$ can simply be obtained by decreasing the terminal's time window by the customer's service time together with the travel time that is needed to move from the customer's location to the terminal. If $i \in \mathrm{IF}_{40}$ denotes an inbound request, then the first visit of a customer is considered by node $r_i^{(s)}$ that represents the

fully loaded container transportation. As a result, the time windows of terminal and customer have to be considered to formulate the time window and the service time of node $r_i^{(s)}$. In order to implement the combined time window, the formula of Zhang et al. [2010], who consider service times, travel times and the offset between the two time windows in their calculation (see also Table 5.1), is used. The second node ($r_i^{(s)}$ for $i \in \mathrm{OF}_{40}$ and $r_i^{(e)}$ for $i \in \mathrm{IF}_{40}$) represents the demand/supply for an empty container arising at the terminal. This node is defined similar to the nodes representing OE/IE hinterland requests.

Fully loaded 20-foot containers do not have to be directly dispatched to their specified destinations (see e.g. the route given by the blue, zigzag arcs in Figure 5.2). In the present graph formulation, three nodes are introduced to model a full 20-foot hinterland request $i \in \mathrm{OF}_{20} \cup \mathrm{IF}_{20}$. The implementation of the node $r_i^{(s)}$ for $i \in \mathrm{OF}_{20}$ and $r_i^{(e)}$ for $i \in \mathrm{IF}_{20}$, which specifies the demand/supply for an empty 20-foot container arising at the terminal, remains the same as in the implementation of 40-foot containers. However, in order to enable detours within the routes of fully loaded 20-foot containers, the previously defined nodes $r_i^{(e)}$ for $i \in \mathrm{OF}_{40}$ and $r_i^{(s)}$ for $i \in \mathrm{IF}_{40}$ decompose into the point of origin ($r_i^{(m)}$ for $i \in \mathrm{OF}_{20}$ and $r_i^{(s)}$ for $i \in \mathrm{IF}_{20}$) and the point of destination ($r_i^{(e)}$ for $i \in \mathrm{OF}_{20}$ and $r_i^{(m)}$ for $i \in \mathrm{IF}_{20}$) and now represent a single (de-)couple operation, instead of an entire container transportation. The corresponding container transportation is represented by the arc that is incident with the two nodes ($(r_i^{(m)}, r_i^{(e)})$ for $i \in \mathrm{OF}_{20}$ and $(r_i^{(s)}, r_i^{(m)})$ for $i \in \mathrm{IF}_{20}$).

5.1.2 (Un-)loading Operations

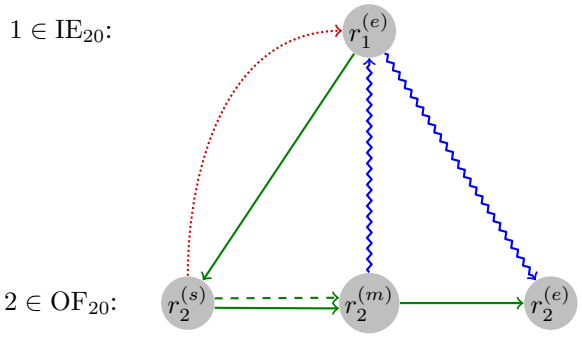


Figure 5.2: Different (de-)coupling possibilities (Funke and Kopfer [2016]).

Figure 5.2 depicts three different possibilities for performing the (de-)coupling operations of two 20-foot container hinterland requests $1 \in \mathrm{IE}_{20}$ and $2 \in \mathrm{OF}_{20}$. The first possibility is shown by solid, green arcs (the parallel dashed arc from $r_2^{(s)}$ to $r_2^{(m)}$ is only introduced to indicate that this arc represents a

loading operation). First of all, the truck performs a street-turn from terminal e_1 to customer s_2 (arc $(r_1^{(e)}, r_2^{(s)})$). Afterwards, the container is loaded at the customer's location in a stay-with procedure (arc $(r_2^{(s)}, r_2^{(m)})$). At the end, the truck carries the fully loaded container from the customer's location to terminal e_2 ($(r_2^{(m)}, r_2^{(e)})$). The second possibility is shown by the dotted, red arc. In this case, the truck decouples an empty container at customer s_2. Afterwards, the container is loaded in a drop-and-pick procedure. This means that arc $(r_2^{(s)}, r_2^{(m)})$ is not used in the presented solution. While the container is loaded, the truck travels to terminal e_1 of the first hinterland request (arc $(r_2^{(s)}, r_1^{(e)})$) at where it collects an empty container. The third possibility applies only to 20-foot container transportation, it is shown by zigzag, blue arcs. Here, the truck does not directly travel from the sender's location s_2 to terminal e_2, which are the point of origin and destination of the same fully loaded container (arc $(r_2^{(m)}, r_2^{(e)})$); the truck detours (arcs $(r_2^{(m)}, r_1^{(e)})$ and $(r_1^{(e)}, r_2^{(e)})$) to collect an empty container obtained at terminal e_1 instead.

Since the majority of locations specify whether or not to (de-)couple containers, the corresponding durations of (de-)coupling operations are included in the service times of the locations' representatives, i.e. the nodes. The only exceptions are (de-)coupling operations, which are performed in the case that a container is (un-)loaded in a drop-and-pick procedure, i.e. since the application of stay-with or drop-and-pick procedure is not determined beforehand, the corresponding (de-)coupling operations are uncertain as well. Within the present model decoupling operations, which are necessary to (un-)load containers in a drop-and-pick procedure, are included in the durations of arcs, while coupling operations, which are necessary to (un-)load containers in a drop-and-pick procedure, are included in the service times of nodes. Table 5.1 reveals time windows, locations and service times that are stored at nodes representing the tasks of hinterland requests. The balance values of nodes are defined in Section 5.2.2.

Let denote by A_L the set of arcs representing (un-)loading operations (the dashed arcs, which are shown in Figure 5.1):

$$\begin{aligned} A_L :=& \{(r_i^{(s)}, r_i^{(e)}) | i \in \mathrm{OF}_{40} \cup \mathrm{IF}_{40}\} \\ & \cup \{(r_i^{(s)}, r_i^{(m)}) | i \in \mathrm{OF}_{20}\} \cup \{(r_i^{(m)}, r_i^{(e)}) | i \in \mathrm{IF}_{20}\} \end{aligned} \tag{5.1}$$

An element of A_L is called loading arc. Apparently, the container is (un-)loaded in a stay-with procedure only when the truck traverses a loading arc. Otherwise, the container is (un-)loaded in a drop-and-pick procedure. As a result, the additional decoupling duration that is needed by the drop-and-pick procedure is added to the duration of all arcs, which leave the tail of an arc in A_L, but do not enter its head.

node v	locations ($\text{orig}_v, \text{dest}_v$)	service time (serv_v)	balance (b_v)	time window ($[\text{start}_v, \text{end}_v]$)
$i \in \text{OF}_{40}$:				
$r_i^{(s)}$	$\text{orig} = \text{dest} = s_i$	0	$b_2 = -1$	$[\text{start}_{s_i}, \text{end}_{s_i}]$
$r_i^{(e)}$	$\text{orig} = s_i, \text{dest} = e_i$	$\text{cpl} + \text{dist}(s_i, e_i) + \text{dcpl}$	$b \equiv 0$	$[\text{start}_{e_i} - \text{dist}(s_i, e_i), \text{end}_{e_i} - \text{dist}(s_i, e_i)]$
$i \in \text{IF}_{40}$:				
$r_i^{(s)}$	$\text{orig} = s_i, \text{dest} = e_i$	$\max\{\text{start}_{e_i} - \text{end}_{s_i}$ $, \text{cpl} + \text{dist}(s_i, e_i)\}$	$b \equiv 0$	$[\min\{\max\{\text{start}_{s_i}, \text{start}_{e_i} - \text{cpl} - \text{dist}(s_i, e_i)\}, \text{end}_{s_i}\}$ $, \min\{\text{end}_{s_i}, \text{end}_{e_i} - \text{cpl} - \text{dist}(s_i, e_i)\}]$
$r_i^{(e)}$	$\text{orig} = \text{dest} = e_i$	cpl	$b_2 = 1$	$[0, H]$
$i \in \text{OF}_{20}$:				
$r_i^{(s)}$	$\text{orig} = \text{dest} = s_i$	0	$b_1 = -1$	$[\text{start}_{s_i}, \text{end}_{s_i}]$
$r_i^{(m)}$	$\text{orig} = \text{dest} = s_i$	cpl	$b_{i+2} = 1$	$[0, H]$
$r_i^{(e)}$	$\text{orig} = \text{dest} = e_i$	dcpl	$b_{i+2} = -1$	$[\text{start}_{e_i}, \text{end}_{e_i}]$
$i \in \text{IF}_{20}$:				
$r_i^{(s)}$	$\text{orig} = \text{dest} = s_i$	cpl	$b_{i+2} = 1$	$[\text{start}_{s_i}, \text{end}_{s_i}]$
$r_i^{(m)}$	$\text{orig} = \text{dest} = e_i$	0	$b_{i+2} = -1$	$[\text{start}_{e_i}, \text{end}_{e_i}]$
$r_i^{(e)}$	$\text{orig} = \text{dest} = e_i$	cpl	$b_1 = 1$	$[0, H]$
$i \in \text{OE}_{40}$:				
$r_i^{(s)}$	$\text{orig} = \text{dest} = s_i$	dcpl	$b_2 = -1$	$[\text{start}_{s_i}, \text{end}_{s_i}]$
$i \in \text{OE}_{20}$:				
$r_i^{(s)}$	$\text{orig} = \text{dest} = s_i$	dcpl	$b_1 = -1$	$[\text{start}_{s_i}, \text{end}_{s_i}]$
$i \in \text{IE}_{40}$:				
$r_i^{(e)}$	$\text{orig} = \text{dest} = s_i$	cpl	$b_2 = 1$	$[\text{start}_{s_i}, \text{end}_{s_i}]$
$i \in \text{IE}_{20}$:				
$r_i^{(e)}$	$\text{orig} = \text{dest} = s_i$	cpl	$b_1 = 1$	$[\text{start}_{s_i}, \text{end}_{s_i}]$

Table 5.1: Implementation of nodes representing requests.

5.1.3 Container Storage Operations

node	duration (serv)	balance (b)	time window ($[\mathrm{start}_v, \mathrm{end}_v]$)
$\{ d_t^+ \mid t \in \mathcal{T} \}$	0	$b \equiv 0$	$[0, H]$
$\{ d_t^- \mid t \in \mathcal{T} \}$	0	$b \equiv 0$	$[0, H]$
d_i^{+40}	cpl	$b_2 = 1$	$[0, H]$
d_i^{+20}	cpl	$b_1 = 1$	$[0, H]$
d_i^{-40}	dcpl	$b_2 = -1$	$[0 - \mathrm{dcpl}, H]$
d_i^{-20}	dcpl	$b_1 = -1$	$[0 - \mathrm{dcpl}, H]$

Table 5.2: Implementation of depot duplicates.

As already mentioned, the present formulation uses the idea of Nossack and Pesch [2013] to implement the several visits of the depot within a truck route. Altogether $|\mathcal{R}| + 2|\mathcal{T}|$ nodes are introduced for the representation of the depot. This composition is explained in the following. Since the problem definition assumes that the depot is able to provide an empty container for each outbound hinterland request and additionally is a sufficiently large stowage for empty containers obtained by inbound hinterland requests, each request introduces one depot duplicate to the node set; in more detail, for each request $i \in \mathrm{OF}_{40} \cup \mathrm{OE}_{40}$ a depot node d_i^{+40}/$i \in \mathrm{OF}_{20} \cup \mathrm{OE}_{20}$ a depot node d_i^{+20}/ $i \in \mathrm{IF}_{40} \cup \mathrm{IE}_{40}$ a depot node d_i^{-40} /$i \in \mathrm{IF}_{20} \cup \mathrm{IE}_{20}$ a depot node d_i^{-20} is introduced. Furthermore, each truck $t \in \mathcal{T}$ introduces two nodes d_t^+ and d_t^-, which serve as the points of origin and destination of the truck's route. A time window that is equal to the time horizon $[0, H]$ is specified for the majority of the depot's duplicates. However, an exemption applies to the time windows of the depot duplicates d_i^{-40}, d_i^{-20} ($i \in \mathcal{R}$) that represent container storage operations taking place at the depot. The reason for the time windows' modification is that the number of depot duplicates representing storage operations exceeds the number of storage operations, which are normally needed by a solution to a problem's instance. Therefore, a solution might contain a container coupling operation immediately preceding a container decoupling operation, which both take place at the depot ($(d_i^{+k}, d_j^{-k}), k \in \{20, 40\}, i, j \in \mathcal{R}$). Since such an operation pair is only included in a solution to the MIP in order to balance the container storage operations, but will not be implemented in a real-world application, these operation pairs should not consume any time. For the simulation of zero duration, depots representing container storage operations are allowed to start dcpl times before the start of the time horizon. Because of (de-)coupling durations being stored at nodes, the duration of arcs

$((d_i^{+k}, d_j^{-k}), k \in \{20, 40\}, i, j \in \mathcal{R})$ is decremented by the sum of the duration needed by (de-)couple operations. Table 5.2 summarizes the considerations for nodes representing the duplicates of the depot.

5.1.4 Entire Graph Representation

arc	duration (dur)	cost (c)
$i \in \mathrm{OF}_{40}, j \in \mathrm{IF}_{40}$:		
(d_i^{+40}, d_j^{-40})	$-\,\mathrm{cpl} - \mathrm{dcpl}$	0
$i \in \mathrm{OF}_{20}, j \in \mathrm{IF}_{20}$:		
(d_i^{+20}, d_j^{-20})	$-\,\mathrm{cpl} - \mathrm{dcpl}$	0
Loading Arcs:		
$e_i = (v, w) \in A_L$	load_i	0
$(v, w'), w' \neq w$	$\mathrm{dcpl} + \mathrm{dist}(\mathrm{dest}_v, \mathrm{orig}_{w'})$	$\mathrm{dist}(\mathrm{dest}_v, \mathrm{orig}_{w'})$
Remaining arcs		
(v, w)	$\mathrm{dist}(\mathrm{dest}_v, \mathrm{orig}_w)$	$\mathrm{dist}(\mathrm{dest}_v, \mathrm{orig}_w)$

Table 5.3: Implementation of arcs.

An arc $(v, w) \in A$ is given two values: a duration dur_{vw} and a cost c_{vw}. Table 5.3 summarizes the different values of arcs. For the reason of truck capacity some of the arcs are removed from the graph, since they cannot be included in any feasible solution. For instance, a truck that carries a 40-foot container must firstly decouple the container before it couples another container. As a result, nodes including a coupling operation of an empty 40-foot container can only be the tail of arcs, whose head is a node that includes a decoupling operation of an empty 40-foot container, and vice versa. Furthermore, it is possible to remove the reverse arcs (w, v) of loading arcs $(v, w) \in A_L$ from the graph.

Figure 5.3 depicts the graph of an instance, which consists of one truck $\mathcal{T} = \{t\}$ and three hinterland requests $\mathcal{R} = (1, 2, 3)$, whereby $\{1, 2\} \subseteq \mathrm{OE}_{40}$ and $\{3\} \subseteq \mathrm{IF}_{40}$. The route of truck t originates from d_t^+ and ends at d_t^-. Node $r_3^{(s)}$ represents the transportation of a fully loaded container from the terminal to the receiver's location. It is possible to obtain empty containers from depot nodes d_1^{+40}, d_2^{+40} or the receiver's node $r_3^{(e)}$. Empty containers are allowed to be delivered to terminal nodes $r_1^{(s)}, r_2^{(s)}$ or depot node d_3^{-40}.

5.2 Assigning Containers

The problem that asks for a construction of flows for containers is modeled as a *Multicommodity Flow Problem (MFP)* (see also Definition 2.21). This problem

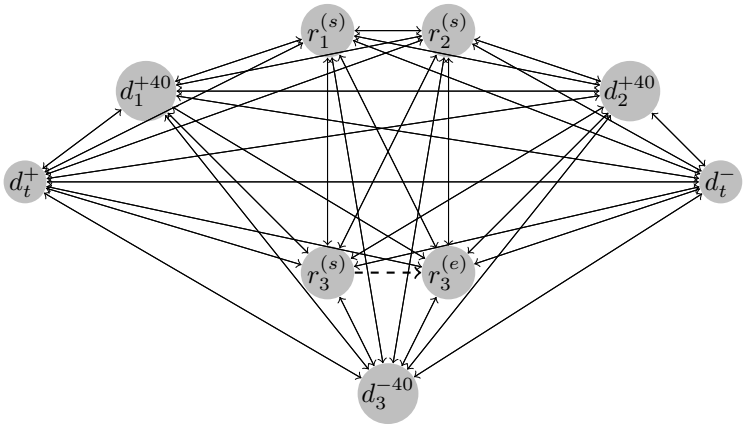


Figure 5.3: Instance graph (Funke and Kopfer [2016]).

assigns empty containers to flexible hinterland requests not only, but also constructs the routes for empty and fully loaded containers. Since the routes for fully loaded 40-foot containers are predetermined (refer to the former section), fully loaded 40-foot containers can be excluded from the route construction process. For a given digraph, the MFP asks for multiple flows, which are indicated by nodes that supply specific commodities to nodes that demand the corresponding commodities. The present formulation uses $2 + |\,\mathrm{IF}_{20} \cup \mathrm{OF}_{20}\,|$ commodities. The meaning of the different commodities is described in more detail in the following. Because of empty containers of the same size being interchangeable (refer to Section 4.1), only one dimension is needed to represent all movements of empty containers sharing a specific size. In the present formulation, the first two dimensions assign the flow for empty 20-foot (first dimension) and empty 40-foot (second dimension) containers to arcs. The remaining $|\,\mathrm{IF}_{20} \cup \mathrm{OF}_{20}\,|$ dimensions are introduced in order to observe that the routes of fully loaded 20-foot containers start and end at their specified locations. A node $v \in V$ is associated with a balance value $b_v^{(k)}$ that expresses the supply/demand of commodity (i.e. container) k. All in all, a node is associated with $2 + |\,\mathrm{IF}_{20} \cup \mathrm{OF}_{20}\,|$ balance values. Since each node is visited exactly once by a truck, an arc can be visited at most once by a truck. Consequently, the arc's capacity u_{vw} limits the total amount of different commodities, which traverse arc $(v, w) \in A$, to the capacity of a truck. For the ease of notation that is used in this section, it is assumed that the set of hinterland requests is ordered $\mathcal{R} := (1, 2, \ldots, |\mathcal{R}|)$ and hinterland requests $\mathrm{IF}_{20} \cup \mathrm{OF}_{20}$ of $\mathcal{R}$ come first among others.

5.2.1 Decision Variables

Overall, three different sets of integral decision variables $x_{vw}^{(1)}, x_{vw}^{(2)}$ and y_{vwi} corresponding to the containers' sizes and filling levels are defined. The set $x_{vw}^{(1)}$ describes the number of empty 20-foot and the set $x_{vw}^{(2)}$ describes the number of empty 40-foot containers traversing arc $(v, w) \in A$, i.e.:

$$\forall k \in \{1,2\}, \forall (v,w) \in A : x_{vw}^{(k)} := |\left\{ c \in \mathcal{C} \,\middle|\, \underset{c}{\text{teu}} = k, c \text{ traverses } (v,w) \right\}| \quad (5.2)$$

Fully loaded 20-foot containers are not interchangeable. For each hinterland request $i \in \mathrm{IF}_{20} \cup \mathrm{OF}_{20}$ let by c_i denote the unique fully loaded container that is to be moved between customer's location and terminal, or vice versa. The decision variables defining the movement of fully loaded 20-foot containers are binary:

$$\forall i \in \mathrm{IF}_{20} \cup \mathrm{OF}_{20}, \forall (v,w) \in A : y_{vwi} := \begin{cases} 1, & \text{if } c_i \text{ traverses } (v,w), \\ 0, & \text{otherwise.} \end{cases} \quad (5.3)$$

5.2.2 Balance Values

For each node a multi-dimensional balance vector b defining the demand/ supply for empty/fully loaded 20-foot and 40-foot containers is introduced. All in all, $2 + |\,\mathrm{IF}_{20} \cup \mathrm{OF}_{20}\,|$ dimensions are considered. The first two dimensions define the demand/supply for empty containers and the remaining dimensions define the demand/supply for fully loaded 20-foot containers, i.e.:

$$\forall v \in V, \forall k \in \{1,2\} : b_v^k := \begin{cases} 1, & v \text{ supplies } c \in \mathcal{C} \text{ with } \mathrm{teu}_c = k, \\ -1, & v \text{ demands } c \in \mathcal{C} \text{ with } \mathrm{teu}_c = k, \\ 0, & \text{otherwise.} \end{cases} \quad (5.4)$$

$$\forall v \in V, \forall i \in \mathrm{IF}_{20} \cup \mathrm{OF}_{20} : b_v^{i+2} := \begin{cases} 1, & v \text{ supplies } c_i, \\ -1, & v \text{ demands } c_i, \\ 0, & \text{otherwise.} \end{cases} \quad (5.5)$$

The distribution of the supply/demand of the different node types is as follows. Since $|\mathcal{R}|$ duplicates of the depot are introduced in order to overcome the imbalances of empty containers caused by the different hinterland requests, these nodes get assigned the complement to the corresponding hinterland request. This means that depot duplicates supply/demand a container, when the associated hinterland request requires/provides a container. In more detail, for $k \in \{20, 40\}$ the depot duplicates d_i^{+k} of outbound requests that are elements of the set $\mathrm{OF}_k \cup \mathrm{OE}_k$ provide empty k-foot containers, while the depot duplicates d_i^{-k} of inbound requests that are elements of the set $\mathrm{IF}_k \cup \mathrm{IE}_k$ demand empty k-foot containers. The reverse balance value is assigned to the

first node $r_i^{(s)}$ of outbound requests $i \in \mathrm{OF} \cup \mathrm{OE}$ or rather the last node $r_i^{(e)}$ of inbound requests $i \in \mathrm{IF} \cup \mathrm{IE}$. For an outbound request $i \in \mathrm{OF}_{20}$, the fully loaded 20-foot container c_i is transported from node $r_i^{(m)}$ that supplies c_i to node $r_i^{(e)}$ that demands c_i. Conversely, for an inbound request $i \in \mathrm{IF}_{20}$, the fully loaded 20-foot container c_i is transported from node $r_i^{(s)}$ that supplies c_i to node $r_i^{(m)}$ that demands c_i. Table 5.1 shows a complete list of balance values. Balance values that are not mentioned in this list are set to zero.

5.2.3 Capacities

An arc $(v, w) \in A$ is traversed by at most one truck. Since each truck is able to carry two TEU the most, the capacity u_{vw} of each arc (v, w) is firstly set to be equal to two. The assumption on the capacity value holds true for arcs that represent moving operations. However, this does not apply to the capacity value of a loading arc $e_i \in A_L$, which represents the (un-)loading operation of container c_i of request $i \in \mathrm{OF} \cup \mathrm{IF}$. This exemption has been made due to container (un-)loading operations being implicitly implemented in the present formulation. This means that there is not assigned any flow representing container c_i to loading arc $e_i \in A_L$. While for trucks it is a matter of optimization whether or not to traverse loading arc e_i, the container c_i must traverse e_i in any case. Therefore, it is sufficient to decrement the capacity of e_i by the size teu_{c_i} of container c_i. Analogously, the transportation of a fully loaded 40-foot container is only implicitly implemented by node $r_i^{(e)}$ (if $i \in \mathrm{OF}_{40}$ is an outbound request) or by node $r_i^{(s)}$ (if $i \in \mathrm{IF}_{40}$ is an inbound request). Because of the capacity of a truck carrying a 40-foot container being fully exploited, only empty trucks are allowed to enter a node representing the transportation of a fully loaded 40-foot container. Overall, it holds:

$$\forall (v, w) \in A : u_{vw} := \begin{cases} 2 - \mathrm{teu}_{c_i}, & (v, w) = e_i \in A_L \\ 0, & w = r_i^{(e)}, i \in \mathrm{OF}_{40} \\ 0, & w = r_i^{(s)}, i \in \mathrm{IF}_{40} \\ 2, & \text{otherwise} \end{cases} \tag{5.6}$$

5.2.4 Entire Model

Figure 5.4 illustrates the balance values of nodes and the capacities of arcs of an instance that comprises three full hinterland requests $\mathcal{R} = (1, 2, 3)$. Hinterland requests $1, 2$ are 20-foot container hinterland requests ($1 \in \mathrm{IF}_{20}, 2 \in \mathrm{OF}_{20}$) and hinterland request 3 is a 40-foot container hinterland request ($3 \in \mathrm{OF}_{40}$). Once more, loading arcs are depicted by dashed arcs. The capacity of loading arcs is decremented by the size of this container, whose (un-)loading operation is

represented by the arc. Each of the two hinterland requests $1, 2 \in \mathrm{IF}_{20} \cup \mathrm{OF}_{20}$ adds an additional dimension to the balance vector in order to ensure that the corresponding fully loaded container is transported between its specified locations leading to a total of four commodities. Moreover, each hinterland request introduces an additional depot. As a result, the depot is able to supply empty containers to outbound requests $2, 3$ and to store the empty container that is provided by inbound request 1. A further possibility for the distribution of empty containers is to perform a street-turn between requests 1 and 2. In this case the solution requires that arc $(r_1^{(e)}, r_2^{(s)})$ gets assigned a flow value greater than zero in the first dimension. Furthermore, if a street-turn is performed, then the arc (d_2^{+20}, d_1^{-20}) gets also assigned a flow value greater than zero. However, this connection does not induce any additional cost or duration, since the service time and the time windows have been adequately decremented and shifted (refer to Tables 5.2 and 5.3). Because of the balance vectors of nodes d_t^+, d_t^-, which serve as the point of origin and destination of a truck t, being equal to zero, these nodes are not depicted in Figure 5.4.

Finally, the MFP can be stated as follows:

$$\sum_{(v,w)\in A} x_{vw}^{(k)} - \sum_{(w,v)\in A} x_{wv}^{(k)} = b_v^{(k)} \qquad \forall v \in V, \forall k \in \{1,2\} \tag{5.7}$$

$$\sum_{(v,w)\in A} y_{vwi} - \sum_{(w,v)\in A} y_{wvi} = b_v^{(i+2)} \qquad \forall v \in V, \forall i \in \mathrm{IF}_{20} \cup \mathrm{OF}_{20} \tag{5.8}$$

$$\sum_{k\in\{1,2\}} \underset{c}{\mathrm{teu}} \cdot x_{vw}^{(k)} + \sum_{i\in \mathrm{IF}_{20}\cup \mathrm{OF}_{20}} y_{vwi} \leq u_{vw} \qquad \forall (v,w) \in A \tag{5.9}$$

$$x_{vw}^{(k)} \in \{0,1,2\} \qquad \forall (v,w) \in A, \forall k \in \{1,2\} \tag{5.10}$$

$$y_{vwi} \in \{0,1\} \qquad \forall (v,w) \in A, \forall i \in \mathrm{IF}_{20} \cup \mathrm{OF}_{20} \tag{5.11}$$

Constraints 5.7 define the supply/demand for empty containers and Constraints 5.8 define the supply/demand for fully loaded 20-foot containers. Constraints 5.9 ensure that that the total flow through an arc does not exceed the arc's capacity. The domains of the sets of decision variables are given by Constraints 5.10 and 5.11.

5.3 Building Routes

The mathematical model that constructs truck routes is build by extending the MIP of an amTSPTW (refer to Sections 2.1.4 and 3.1.1). One of the reasons for the problem being asymmetric is that the transportation of fully loaded 40-foot containers is implemented by a single node $v \in V$, for which the locations orig_v and dest_v differ from one another. Each truck $t \in \mathcal{T}$ adds two depot duplicates d_t^+ and d_t^- to the node set of the graph. Truck t has to start its route from d_t^+

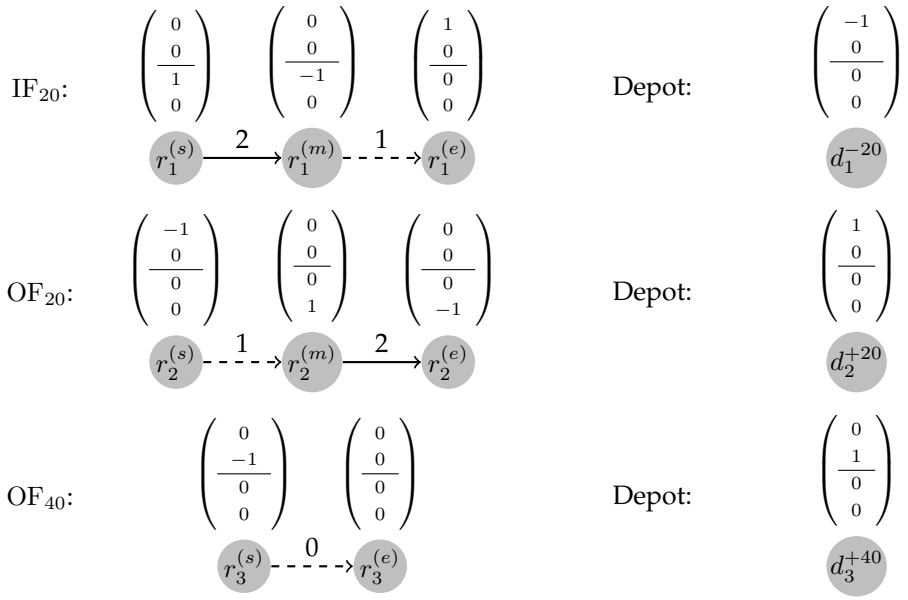


Figure 5.4: Balance values and capacities (Funke and Kopfer [2016]).

and end its route at d_t^- within the time horizon. The remaining tasks have to be assigned to truck routes. Afterwards, an order of the assigned tasks has to be determined resulting in trucks arriving at the corresponding locations within the time windows of the locations.

5.3.1 Decision Variables

Almost all of the nodes are exactly once entered and left by a truck. The only exceptions are the depot duplicates d_t^+, d_t^- ($t \in \mathcal{T}$), which represent the points of origin and destination of the route of truck t. While node d_t^+ is left only but not entered, the opposite applies to node d_t^-. A continuous decision variable t_v is introduced for each node v. The variable t_v obtains the value of the point in time location orig_v is visited by a truck:

$$v \in V : t_v = \{\, a \in \mathbb{R} \,|\, \text{point in time a truck visits } v \,\} \tag{5.12}$$

A further binary decision variable δ_{vw} is introduced for each arc $(v, w) \in A$ in order to determine whether or not a truck traverses arc (v, w):

$$\delta_{vw} := \begin{cases} 1, & \text{if a truck traverses arc } (v, w), \\ 0, & \text{otherwise.} \end{cases} \tag{5.13}$$

5.3.2 Entire Model

The MIP for the construction of truck routes can be build as follows:

$$\sum_{(v,w)\in A} \delta_{vw} = \sum_{(w,v)\in A} \delta_{wv} = 1 \qquad \forall v \in V \setminus \left\{ d_t^+, d_t^- \,\middle|\, t \in \mathcal{T} \right\} \tag{5.14}$$

$$\sum_{(v,w)\in A} \delta_{vw} = 1, \sum_{(w,v)\in A} \delta_{wv} = 0 \qquad \forall v \in \left\{ d_t^+ \,\middle|\, t \in \mathcal{T} \right\} \tag{5.15}$$

$$\sum_{(v,w)\in A} \delta_{vw} = 0, \sum_{(w,v)\in A} \delta_{wv} = 1 \qquad \forall v \in \left\{ d_t^- \,\middle|\, t \in \mathcal{T} \right\} \tag{5.16}$$

$$\delta_{vw} = 1 \Rightarrow t_v + \underset{vw}{\mathrm{dur}} + \underset{v}{\mathrm{serv}} \leq t_w \qquad \forall (v,w) \in A \setminus A_L \tag{5.17}$$

$$\underset{v}{\mathrm{start}} \leq t_v \leq \underset{v}{\mathrm{end}} \qquad \forall v \in V \setminus \left\{ r_i^{(e)} \,\middle|\, i \in \mathrm{OF}_{40} \right\} \tag{5.18}$$

$$t_v \in \mathbb{R} \qquad \forall v \in V \tag{5.19}$$

$$\delta_{vw} \in \{0,1\} \qquad \forall (v,w) \in A \tag{5.20}$$

Constraints 5.14 ensure that the flow conservation rule holds for all nodes except the depot duplicates, which serve as the point of start and end of the truck routes. Constraints 5.15 and 5.16 ensure that starting/ending depot nodes are left/entered exactly once by a truck. Sub-tours are eliminated and time windows are set by Constraints 5.17 and 5.18 (the reason for excluding nodes $r_i^{(e)}, i \in \mathrm{OF}_{40}$ from Constraints 5.18 is stated in the next section). Finally, Constraints 5.19 and 5.20 set the domains of the sets of decision variables.

5.4 Coupling of the Models

For the purpose of combining the mathematical models of the two sub-problems of the mICTP, two necessary conditions have to be considered. On the one hand, containers constitute passive vehicles that must be carried by a truck to move in space (Drexl [2012], see also Section 3.1.2). This means that all container flows have to be covered by the truck routes. On the other hand, the decision whether to apply stay-with or drop-and-pick procedure to the different (un-)loading operations has not yet been taken. Up to now, container (un-)loading operations have been only implicitly modeled by decreasing the capacity u_{e_i} of loading arc $e_i \in A_L$ by the container's size teu_{c_i}. The advantage of this subtraction is that no extra flow representing c_i has to be assigned to arc e_i. Since not any flow representing c_i traverses arc e_i, truck routes may, but need not, cover arc e_i. The decision making process determines, whether trucks traverse arc e_i (stay-with procedure) or not (drop-and-pick procedure). The additional duration that is needed by (un-)loading and (de-)coupling op-

erations has to be integrated into the model as well.

$$\sum_{k\in\{1,2\}} k \cdot x_{vw}^{(k)} + \sum_{i\in \mathrm{IF}_{20}\,\cup\,\mathrm{OF}_{20}} y_{vwi} \leq 2 \cdot \delta_{vw} \qquad \forall (v,w) \in A \tag{5.21}$$

$$\delta_{vw} = 0 \Rightarrow t_v + \mathrm{dcpl} + \underset{v}{\mathrm{serv}} + \underset{vw}{\mathrm{dur}} \leq t_w \qquad \forall (v,w) \in A_L \tag{5.22}$$

$$\delta_{vw} = 1 \Rightarrow t_v + \underset{vw}{\mathrm{dur}} + \underset{v}{\mathrm{serv}} - \mathrm{cpl} \leq t_w \qquad \forall (v,w) \in A_L \tag{5.23}$$

$$\underset{r_i^{(e)}}{\mathrm{start}} -(1 - \delta_{r_i^{(s)} r_i^{(e)}}) \cdot \mathrm{cpl} \leq t_{r_i^{(e)}} \qquad \forall i \in \mathrm{OF}_{40} \tag{5.24}$$

$$t_{r_i^{(e)}} \leq \underset{r_i^{(e)}}{\mathrm{end}} -(1 - \delta_{r_i^{(s)} r_i^{(e)}}) \cdot \mathrm{cpl} \qquad \forall i \in \mathrm{OF}_{40} \tag{5.25}$$

Constraints 5.22 and 5.23 can be combined to form one constraint:

$$t_v + (1 - \delta_{vw}) \cdot \mathrm{dcpl} + \underset{vw}{\mathrm{dur}} + \underset{v}{\mathrm{serv}} - \delta_{vw} \cdot \mathrm{cpl} \leq t_w \quad \forall (v,w) \in A_L \tag{5.26}$$

Constraints 5.21 ensure that each container flow is covered by a part of a truck's route. The implementation of constraints representing the (un-)loading operations of containers is inspired by Goel and Meisel [2013], who provide a mT-SPTW formulation of a maintenance problem of electricity networks (see also Section 3.1.2). As already mentioned, if a loading arc $e_i = (v,w) \in A_L$ of a full hinterland request $i \in \mathrm{OF} \cup \mathrm{IF}$ is not traversed by any truck (Constraints 5.22), then the fully loaded container c_i is (un-)loaded in a drop-and-pick procedure. This means that some precedence time has to take place between the starting times of nodes v and w. This precedence time ensures that the former task (i.e. decoupling and (un)-loading the container) has been completed before the latter (i.e. collecting the container) is allowed to start. Conversely, if a truck traverses loading arc $e_i \in A_L$, then c_i is (un-)loaded in a stay-with procedure. Constraints 5.23 are build almost in the same way as Constraints 5.17, apart from one minor exception: since the service time of node w includes the duration that is needed to couple a container, the precedence time has to be decreased by this coupling duration, if a container is (un-)loaded in a drop-and-pick procedure. Because of the assumption that the second visit of a customer is not restricted to any time window, this virtual shift by the container's coupling duration has no impact on nodes representing customers' locations. However, the node $r_i^{(e)}$ of a 40-foot container OF request $i \in \mathrm{OF}_{40}$ represents a customer's location not only, but also implements the transportation of a fully loaded container from the sender's location to the terminal. Therefore, the arrival time of a truck at node $r_i^{(e)}, i \in \mathrm{OF}_{40}$ has to depend on the terminal's time window as well. As a result, Constraints 5.24 and 5.25 set the different time windows of nodes $r_i^{(e)}, i \in \mathrm{OF}_{40}$, in the case that stay-with or drop-and-pick procedure is performed.

5.5 Objectives

The computational study that is provided in Chapter 7 considers two different objectives. The study focuses especially on the objective that minimizes the total travel distance of the trucks:

$$\min \sum_{e \in A} c_e \cdot \delta_e + \sum_{i \in \mathrm{OF}_{40}} \mathrm{dist}(\underset{r_i^{(e)}}{\mathrm{orig}}, \underset{r_i^{(e)}}{\mathrm{dest}}) + \sum_{i \in \mathrm{IF}_{40}} \mathrm{dist}(\underset{r_i^{(s)}}{\mathrm{orig}}, \underset{r_i^{(s)}}{\mathrm{dest}}) \tag{5.27}$$

The first summand of Objective 5.27 represents the actual objective, while the last two summands are constants, which represent the carriage of fully loaded 40-foot containers.

The second objective minimizes the total operating time of the trucks:

$$\min \sum_{v \in \{\, d_t^- \,\mid t \in \mathcal{T} \,\}} t_v - \sum_{v \in \{\, d_t^+ \,\mid t \in \mathcal{T} \,\}} t_v \tag{5.28}$$

Objective 5.28 decrements the sum of the times trucks enter ending depots by the sum of the times trucks leave starting depots.

Chapter 6

Heuristic Approach

In order to solve the mICTP, a matheuristic is developed; the basic idea of the approach is that the two phases that assign empty containers to flexible tasks and construct routes for trucks are combined in one *Large Neighborhood Search (LNS)* (Ropke and Pisinger [2006]). Among others, Braekers et al. [2013] investigate heuristic approaches consisting of these two phases. In their study Braekers et al. [2013] examine that the approach, which sequentially performs the two phases, is clearly outperformed by an integrated approach. To overcome these results, both phases are strongly interlinked in the present heuristic approach. In addition, one single iteration of the heuristic approach is permitted to take up much computation time. This strategy is inspired by Ropke and Pisinger [2006] that allow large moves in one single iteration to obtain diversification in their approach. Altogether, linear and integer programming models assign empty containers to flexible tasks and thereby compute well-defined tasks. Afterwards, numerous different heuristics construct routes for trucks out of the obtained well-defined tasks. The heuristic approach is based on the neighborhood search of Funke and Kopfer [2015], who investigate a restriction of the mICTP that does not consider time windows. Compared to the approach of Funke and Kopfer [2015], the presented heuristic approach can not only be applied to a problem definition that considers time windows, but also integrates several recent destroy and repair strategies as well as another mechanism for the update of current solutions, i.e. the hill climbing strategy is replaced by SA (refer to Section 2.2.3).

This chapter introduces names for each parameter that is contained in the heuristic approach. The introduction of names leads to a simplified referencing to parameters in Section 7.4.1, which addresses the choice of values for the parameters. The remainder of this chapter is organized as follows: Section 6.1 introduces the models used to assign containers and Section 6.2 introduces the

basic structure of the heuristics used to build truck routes. The entire heuristic approach approach is presented in Section 6.3 and implementation details to speed up computation time are listed in Section 6.4.

6.1 Models for Containers

In order to construct routes comprising flexible tasks that have not specified the point of origin or destination, empty containers are assigned to flexible tasks before. The assignment of empty containers to flexible tasks indirectly assigns the missing locations to flexible tasks, since the point of pickup and delivery is known for empty containers. In this step, all pickup and delivery tasks of containers are combined to form compound container transportation requests. A compound container transportation request constitutes a minimum connected component of a truck's route that is not separable because at least one container is shared between the different pickup and delivery tasks. Meisel and Kopfer [2014] take a similar approach by constructing *carriages*, i.e. sub-tours of non-autonomous recourses (refer to Section 3.3.2). Altogether, three different linear and integer programming models for the exact computation of *connected components* are presented in the following.

6.1.1 Connected Components

The chosen representation of hinterland requests in this chapter differs from the representation that is used in Chapter 5. Hinterland requests are divided into container pickup ("$\oplus$") and delivery ("$\ominus$") tasks. Figure 6.1 depicts the different container pickup and delivery tasks of the different hinterland requests that are contained in the mICTP. Similar to the representation that is used in Chapter 5, OF and IF hinterland requests decompose into three different tasks, while OE and IE hinterland requests consist of one single task. OF and IF hinterland requests are subdivided into one well-defined task representing a fully loaded container transportation (shown by solid lined rectangles) and one flexible task representing a pickup or delivery request of an empty container (shown by dashed lined rectangles). The composition of an OF hinterland request is described here as an example of both full hinterland requests. The dashed lined rectangle at the beginning represents the sender's demand for an empty container, while the subsequent solid lined rectangle represents the transportation of the fully loaded container from the sender's location to the terminal. The customers' locations of OF and IF requests again are duplicated in this representation: one duplicate refers to the empty container handling operation and the other duplicate refers to the fully loaded container handling

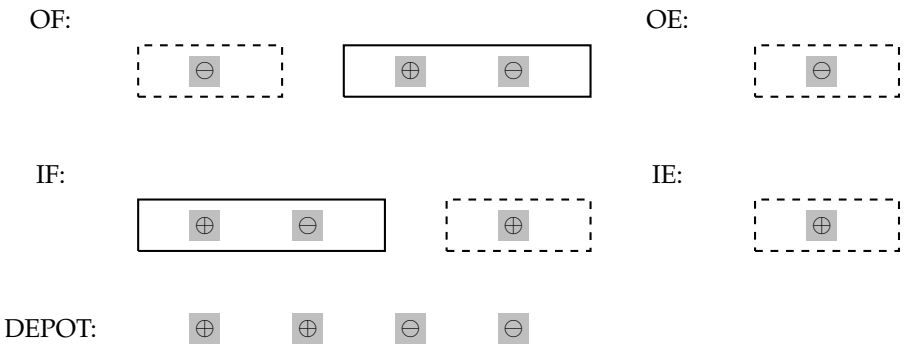


Figure 6.1: Pickup and delivery tasks (Funke and Kopfer [2015]).

operation (see also Chapter 5). For OE and IE hinterland requests, one dashed lined rectangle is used to represent the flexible delivery or pickup task. Finally, a set of individual flexible pickup and delivery tasks is introduced for the representation of the stowage operations of empty containers taking place at the depot. To achieve a mathematical definition of connected components, sets $\mathcal{E}_k$, $\mathcal{D}_k$ and $\mathcal{F}_k$ for $k \in \{20, 40\}$ are defined as follows:

$$\forall k \in \{20, 40\}: \quad \mathcal{E}_k := \left\{ \ominus_r \,\middle|\, r \in \underset{k}{\mathrm{OF}} \cup \underset{k}{\mathrm{OE}} \right\} \cup \left\{ \oplus_r \,\middle|\, r \in \underset{k}{\mathrm{IF}} \cup \underset{k}{\mathrm{IE}} \right\} \tag{6.1}$$

$$\forall k \in \{20, 40\}: \quad \mathcal{D}_k := \left\{ \oplus_r \,\middle|\, r \in \underset{k}{\mathrm{OF}} \cup \underset{k}{\mathrm{OE}} \right\} \cup \left\{ \ominus_r \,\middle|\, r \in \underset{k}{\mathrm{IF}} \cup \underset{k}{\mathrm{IE}} \right\} \tag{6.2}$$

$$\forall k \in \{20, 40\}: \quad \mathcal{F}_k := \left\{ \oplus_r, \ominus_r \,\middle|\, r \in \underset{k}{\mathrm{OF}} \cup \underset{k}{\mathrm{IF}} \right\} \tag{6.3}$$

The set $\mathcal{E}_k$ contains the flexible pickup and delivery tasks of empty k-foot container hinterland requests, i.e. the dashed lined rectangles that are used in the representation of hinterland requests in Figure 6.1. The set $\mathcal{D}_k$ represents the pickup and delivery tasks of empty k-foot containers taking place at the depot that are depicted at the bottom of Figure 6.1. The set $\mathcal{F}_k$ contains the well-defined pickup and delivery tasks of fully loaded k-foot container hinterland requests, i.e. the solid lined rectangles that are used in the representation of hinterland requests in Figure 6.1. Given these definitions, a connected component can be defined as follows:

Definition 6.1 (Connected Component)**.** A sequence $(l_1, \ldots, l_n) \subseteq \mathcal{E}_k \cup \mathcal{D}_k \cup \mathcal{F}_k$, $k \in \{20, 40\}$, is called **connected component**, when it holds

$$\forall j \in \{1, \ldots, n-1\} : 0 < \sum_{i=1}^{j} \mathrm{cap}(i) \leq 2 \text{ and } \sum_{i=1}^{n} \mathrm{cap}(i) = 0,$$

where $\mathrm{cap}(i)$ denotes the size (measured in TEU) of the in l_i coupled or decoupled container, if l_i is a pickup task; $\mathrm{cap}(i)$ denotes minus the size of the container, if l_i is a delivery task.

Let $\mathcal{P}$ state the problem that asks for a construction of connected components. A truck's capacity leads to three different possibilities for coupling containers at one point in time: nothing, 40-foot containers only or 20-foot containers only. Hence, $\mathcal{P}$ can be partitioned into $\mathcal{P} = \mathcal{P}_{40} \dot{\cup} \mathcal{P}_{20}$ without influencing the solution quality.

6.1.2 Graph

Since 40-foot and 20-foot containers can separately be assigned, two different graph representations $G_k = (V_k, A_k)$ are introduced for the different container sizes $k \in \{20, 40\}$. Nodes are introduced for each pickup and delivery task that is contained in the set $\mathcal{D}_k \cup \mathcal{E}_k$. In contrast to the transportation of 40-foot containers, trucks are able to simultaneously carry fully loaded and empty 20-foot containers. For this reason, the node set V_{20} additionally includes elements that are contained in the set $\mathcal{F}_{20}$. Arcs are introduced between two nodes whenever it is possible for a truck to move between the different pickup and delivery tasks represented by the nodes (i.e. the time windows of different locations need to be considered as it is described in the next paragraph). For a node $v_i \in V_k$ let the corresponding pickup or delivery task be denoted by $l_i \in \mathcal{D}_k \cup \mathcal{E}_k \cup \mathcal{F}_k$. We define mappings associated to nodes and arcs. Each node v_i is associated with the location loc_i where l_i takes place and the time interval $[\mathrm{start}_i, \mathrm{end}_i]$ in which l_i has to start; v_i gets assigned a balance value $b_i := \mathrm{cap}(i)$ and a service time serv_i that states the duration of l_i, i.e. $\mathrm{serv}_i := \mathrm{cpl}$, if l_i is a pickup task $\oplus$ and $\mathrm{serv}_i := \mathrm{dcpl}$, if l_i is a delivery task $\ominus$. For each arc $(v_i, v_j) \in A_{40} \cup A_{20}$, the value $\mathrm{dist}_{i,j}$ is set to the time needed to travel from loc_i to loc_j. Given these notations, the different sets of nodes and arcs can be defined as follows:

$$V_{20} := \{\, v_i \,|\, l_i \in \mathcal{E}_{20} \cup \mathcal{D}_{20} \cup \mathcal{F}_{20} \,\}, V_{40} := \{\, v_i \,|\, l_i \in \mathcal{E}_{40} \cup \mathcal{D}_{40} \,\}$$
$$\forall k \in \{20, 40\} : A_k := \left\{ (v_i, v_j) \in V_k \times V_k \,\middle|\, \underset{i}{\mathrm{start}} + \underset{i,j}{\mathrm{dist}} + \underset{i}{\mathrm{serv}} - \underset{j}{\mathrm{end}} \leq 0 \right\} \tag{6.4}$$

If $(\mathrm{start}_i + \mathrm{dist}_{i,j} + \mathrm{serv}_i - \mathrm{end}_j) > 0$ holds true, then time windows between two pickup/delivery tasks l_i and l_j cannot be met. Therefore, the corresponding arcs are excluded from graphs.

Similar to the container assignment problem that is solved by the exact approach (refer to Section 5.2), balance values become multidimensional, when

the simultaneous transportation of empty and fully loaded 20-foot containers is considered. Each transportation of a fully loaded container induces one dimension to the balance vector leading to a mapping $b_i^{(p)}$ with $p \in [\dim]$, whereby $\dim := \frac{|\mathcal{F}_{20}|}{2} + 1$ (refer to Definition 2.2). The additional dimensions are introduced to observe the given points of origin and destination of a fully loaded container. Table 6.1 shows a listing of the different balance values, not mentioned values are set to zero.

node v_i	$\mathcal{E}_{40} \cup \mathcal{D}_{40}$	$\mathcal{E}_{20} \cup \mathcal{D}_{20}$	$\mathcal{F}_{20}$
$l_i \in \oplus$	$b_i = 1$	$b_i^{(1)} = 1$	$b_i^{(\lceil \frac{i}{2} \rceil + 1)} = 1$
$l_i \in \ominus$	$b_i = -1$	$b_i^{(1)} = -1$	$b_i^{(\lceil \frac{i}{2} \rceil + 1)} = -1$

Table 6.1: Different balance values of pickup and delivery tasks.

6.1.3 Assignment of Empty Containers

We first consider the assignment of empty containers only, i.e. for the remainder of this section $\mathcal{F}_k := \emptyset$ holds for $k \in \{20, 40\}$. In this case, the two subproblems $\mathcal{P}_{40}$ and $\mathcal{P}_{20}$ of assigning empty containers to flexible tasks can be transformed to well-known combinatorial optimization problems.

Assignment of Empty 40-foot Containers

The assignment of empty 40-foot containers is already discussed in scientific sources. Nossack and Pesch [2013] show that $\mathcal{P}_{40}$ reduces to a classical assignment problem to which optimum solutions can be computed in polynomial time (see also Remark 2.15). The node set V_{40} is divided into two partitions $V_{40}^{\oplus} \dot{\cup} V_{40}^{\ominus}$; partition $V_{40}^{\oplus}$ contains nodes representing pickup tasks and partition $V_{40}^{\ominus}$ contains nodes representing delivery tasks. It follows from the definitions of the sets $\mathcal{D}_{40}$ and $\mathcal{E}_{40}$ that $V_{40}^{\oplus}$ and $V_{40}^{\ominus}$ are sets of the same cardinality. The problem $\mathcal{P}_{40}$ can be stated as follows:

$$\min \sum_{(v_i,v_j) \in (A_{40} \cap (V_{40}^{\oplus} \times V_{40}^{\ominus}))} (\underset{i,j}{\text{dist}} + \max\{\underset{j}{\text{start}} - \underset{i}{\text{end}} - \underset{i,j}{\text{dist}} - \underset{i}{\text{serv}}, 0\}) \cdot x_{ij} \quad (6.5)$$

$$\text{s.t.} \sum_{v_j \in V_{40}^{\ominus} : (v_i,v_j) \in \delta_{v_i}^{+}} x_{ij} = 1 \qquad \forall v_i \in V_{40}^{\oplus} \quad (6.6)$$

$$\sum_{v_i \in V_{40}^{\oplus} : (v_i,v_j) \in \delta_{v_j}^{-}} x_{ij} = 1 \qquad \forall v_j \in V_{40}^{\ominus} \quad (6.7)$$

$$x_{ij} \in \{0,1\} \qquad \forall (v_i,v_j) \in (A_{40} \cap (V_{40}^{\oplus} \times V_{40}^{\ominus})) \quad (6.8)$$

For each arc $(v_i, v_j) \in (A_{40} \cap (V_{40}^{\oplus} \times V_{40}^{\ominus}))$ a binary decision variable x_{ij} is introduced. The decision variable specifies whether or not a container is assigned

between a pair of pickup and delivery tasks (Constraints 6.8). The Objective function 6.5 minimizes the sum of the total travel time and total waiting time. Constrains 6.6 - 6.7 ensure that each task is exactly once assigned.

Assignment of Empty 20-foot Containers

Empty 20-foot containers are assigned by a variation of the MCFP . However, in contrast to solving the MCFP (refer to Remark 2.20), computing solutions to $\mathcal{P}_{20}$ is $\mathcal{NP}$-hard. The complexity of the variation is shown in the subsequent section. For each arc $(v_i, v_j) \in A_{20}$ an integral decision variable $x_{ij} \in \{0, 1, 2\}$ is introduced. The value of the decision variable specifies the container flow, i.e. the number of empty 20-foot containers moving from loc_i to loc_j. The goal of the MCFP is to minimize the total amount of container flow by routing all amount of container flow obtained by nodes having positive assigned balance value to nodes having negative assigned balance value. The classical MCFP formulation is modified by the introduction of another decision variable z_{ij}. The additional decision variable limits the number of the in-going and the out-going flow arcs of each node to one:

$$\min \sum_{(v_i,v_j)\in A_{20}} (\underset{i,j}{\mathrm{dist}} + \max\{\underset{j}{\mathrm{start}} - \underset{i}{\mathrm{end}} - \underset{i,j}{\mathrm{dist}} - \underset{i}{\mathrm{serv}}, 0\}) \cdot z_{ij} \tag{6.9}$$

$$\text{s.t.} \sum_{(v_i,v_j)\in\delta^+_{v_i}} x_{ij} - \sum_{(v_j,v_i)\in\delta^-_{v_i}} x_{ji} = b_i \qquad \forall v_i \in V_{20} \tag{6.10}$$

$$x_{ij} \leq 2 \cdot z_{ij} \qquad \forall (v_i, v_j) \in A_{20} \tag{6.11}$$

$$z_{ij} \leq x_{ij} \qquad \forall (v_i, v_j) \in A_{20} \tag{6.12}$$

$$\sum_{(v_i,v_j)\in\delta^+_{v_i}} z_{ij} \leq 1 \qquad \forall v_i \in V_{20} \tag{6.13}$$

$$\sum_{(v_i,v_j)\in\delta^-_{v_j}} z_{ij} \leq 1 \qquad \forall v_j \in V_{20} \tag{6.14}$$

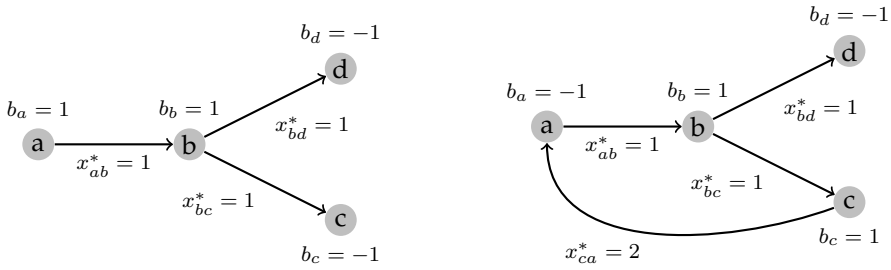
$$x_{ij} \in \{0, 1, 2\} \qquad \forall (v_i, v_j) \in A_{20} \tag{6.15}$$

$$z_{ij} \in \{0, 1\} \qquad \forall (v_i, v_j) \in A_{20} \tag{6.16}$$

The Objective function 6.9 minimizes the sum of the total travel time and the total waiting time. Constraints 6.10 set the flow balance conservation. Constraints 6.11 and 6.12 set the binary decision variable z_{ij} to one, if the value of decision variable x_{ij} is greater than zero; the decision variable z_{ij} is set to zero, otherwise. The maximum number of the out-going arcs and the in-going arcs of a node are restricted by one by Constrains 6.13 and 6.14. Finally, Constraints 6.15 and 6.16 define the value domains of the decision variables x and z.

It is assumed that the costs of arcs are greater than zero in this section. A variation of the MCFP has to be considered for the following reason. For

$(v_i, v_j) \in A_{20}$ let x^*_{ij} denote an optimum solution to an MCFP instance. We consider the graph $G^*_{20} := (V_{20}, A^*_{20})$ with $A^*_{20} := \{ (v_i, v_j) \in A_{20} \,|\, x^*_{ij} > 0 \}$ that contains arcs having assigned a positive flow value only. In a solution to $\mathcal{P}_{20}$, the connected components in G^*_{20} (see Definition 2.7) correspond to the minimum connected components of a truck's route (refer to Definition 6.1). However, in order to construct unique truck routes, the connected components of G^*_{20} have to be paths. This property need not apply to every optimum MCFP solution, as shown in Figure 6.2. At the left a solution, which contains one



(a) Positive Costs. **(b)** Costs greater or equal to zero.

Figure 6.2: Characteristic of connected components.

connected component that is no path, is depicted. The sequence, in which the locations corresponding to nodes c and d are visited, is not determined in this example. Consequently, the number of the in-going and the out-going flow arcs are limited to one in $\mathcal{P}_{20}$ leading to a variation of the MCFP.

Complexity of Assigning Empty 20-Foot Containers

By Remark 2.20, an optimum integral solution to the MCFP can be computed in polynomial time. Nevertheless, the variation of the MCFP contains a decision problem that is proofed to be $\mathcal{NP}$-hard; this means that in contrast to the assignment problem of empty 40-foot containers the assignment problem of empty 20-foot containers is $\mathcal{NP}$-hard, regardless of whether or not time windows are considered. The proof of the complexity of $\mathcal{P}_{20}$ follows from the complexity of the *Arc Partition Minimum Cost Flow Problem (APMCFP)*. Eiglsperger [2003] defines the APMCFP that states a generalization of the MCFP. An instance of the APMCFP differs from an instance of the MCFP in that arcs are given no capacities. Instead, the arc set is partitioned into so-called *devices* $D := \{ d \,|\, d \subseteq A \}$, such that $A := \dot{\cup}_{d \in D}\ d$. Each device is given a capacity $u : D \to \mathbb{R}^+$. Moreover, the Capacity Constraints 2.7 of the MCFP are replaced

by the *device capacity constraints* in the APMCFP:

$$\forall d \in D : \sum_{e \in d} f(e) \leq u_d \tag{6.17}$$

Eiglsperger [2003] proofs that computing an optimum integral solution to an APMCFP instance is $\mathcal{NP}$-hard, even in the case that the maximum number of arcs in a device is bounded by two and the maximum capacity of devices is one. In other words, deciding which of two arcs is allowed to get assigned a flow value greater than zero is $\mathcal{NP}$-hard. Constraints 6.13 and 6.14 ask for the assignments of flow values to devices $\{ \delta_v^+ \mid v \in V \}$ and $\{ \delta_v^- \mid v \in V \}$ that are given a capacity function equal to one. This decision problem includes the decision problem, asking for the assignment of a flow value greater than zero to one of two arcs. As a result, it follows that computing an optimum solution to $\mathcal{P}_{20}$ is $\mathcal{NP}$-hard.

6.1.4 Assignment of Empty and Fully Loaded 20-foot Containers

If $\mathcal{F}_{20} \neq \emptyset$ holds true, then connected components that include the pickup and the delivery tasks of empty and fully loaded 20-foot containers have to be constructed. The resulting problem can be formulated as MFP that is known to be $\mathcal{NP}$-hard (see also Remark 2.22). An integral decision variable $x_{ij}^{(p)} \in \{0, 1, 2\}$ is introduced for each arc $(v_i, v_j) \in A_{20}$ and each commodity $p \in [\dim]$. This decision variable specifies the number of 20-foot containers moving from loc_i to loc_j not only, but also determines, which type of container is transported, i.e. $x_{ij}^{(1)} \neq 0$ for empty container transportation or $x_{ij}^{(p)} \neq 0, 2 \leq p \leq \dim$ for the transportation of a specific fully loaded container. The goal is to minimize the total amount of container flow by routing all amount of the container flow of the different commodities obtained by nodes having assigned a positive balance value in the corresponding dimension to nodes that have assigned a negative balance value in the corresponding dimension. Therefore, Objective function 6.9, Constraints 6.13, 6.14 and 6.16 remain the same. Only Constraints 6.10,6.11, 6.12 and 6.15 have to adapted by the following constraint set to consider more than one dimension:

$$\sum_{(v_i,v_j)\in\delta^+_{v_i}} x^{(p)}_{ij} - \sum_{(v_j,v_i)\in\delta^-_{v_i}} x^{(p)}_{ji} = b^{(p)}_i \qquad \forall v_i \in V_{20}, \forall p \in [\dim] \tag{6.18}$$

$$\sum_{p\in[\dim]} x^{(p)}_{ij} \leq 2 \cdot z_{ij} \qquad \forall (v_i,v_j) \in A_{20} \tag{6.19}$$

$$z_{ij} \leq \sum_{p\in[\dim]} x^{(p)}_{ij} \qquad \forall (v_i,v_j) \in A_{20} \tag{6.20}$$

$$x^{(p)}_{ij} \in \{0,1,2\} \qquad \forall (v_i,v_j) \in A_{20}, \forall p \in [\dim] \tag{6.21}$$

By analogy with the Model $6.9 - 6.16$, the connected components (Definition 6.1) of a truck's route can be deduced from the connected components (Definition 2.7) of the MFP graph, in which arcs having assigned a zero flow value are removed.

Iterations and Diversification

The three different models that compute connected components are invoked several times within one call of the heuristic approach. However, several calls of deterministic algorithms do not lead to different solutions. For this reason, two techniques that diversify the solution space are introduced. The heuristic approach randomly decides whether or not to use one of the two techniques.

The first technique is particularly effective in contexts in which solutions are computed to small-sized and middle-sized instances. A tabu list stores recently visited arcs, i.e. arcs $(v_i, v_j) \in A_{40} \cup A_{20}$ with $x_{ij} > 0 / x^{(p)}_{ij} > 0$ for one $p \in [\dim]$. Arcs that appear in the tabu list get assigned the length of the horizon H as cost value in the Objective function 6.5/6.9 for the next executions of the corresponding model. The penalization of recently visited arcs prevents the heuristic algorithm from multiple visits of the same solution within the solution space. If the tabu tenure extends a certain value $\xi \in \mathbb{N}$, then arcs are removed from the tabu list in a FIFO technique.

The second technique is particularly effective if solutions are computed to large-sized instances. A very recent technique that is included in the ALNS of Ropke and Pisinger [2006] is to modify the objective function for some iterations; instead of considering actual costs, a random number *noise* is added to the objective function. This means that the value $\mathrm{dist}_{i,j}$, which denotes the cost of arc $(v_i, v_j) \in A_{40} \cup A_{20}$, changes to $\widetilde{\mathrm{dist}}_{i,j} = \max\{0, \mathrm{dist}_{i,j} + no\}$ in the Objective function 6.5/6.9, whereby $no \in \mathbb{R}$ (the noise) is a randomly chosen number in the interval $[-N_{\max}, N_{\max}]$ with $N_{\max}$ is set to $\eta \cdot \max_{(v_i,v_j)\in A_{20}} \mathrm{dist}_{i,j}$. In the computation of $N_{\max}$, the parameter η controls the amount of noise. In order to achieve the intensification of the local search in late steps of the heuristic algorithm, the value of η decreases during the search process. The change from

function dist to function $\widetilde{\mathrm{dist}}$ may assign a zero cost value to some of the arcs. Optimum solutions to MCFP as well as MFP instances, which contain arcs having assigned a zero cost value, may include cycles (refer e.g. to the right of Figure 6.2). Hence, integral decision variables m and the subtour elimination constraints of Miller et al. [1960] are added to the Models 6.9 – 6.16 and 6.18 – 6.21 in order to exclude cycles from the solution space:

$$m_i + |V_{20}| \cdot z_{ij} - m_j \leq |V_{20}| - 1 \qquad \forall (v_i, v_j) \in A_{20} \tag{6.22}$$

$$m_i \in \mathbb{N} \qquad \forall v_i \in V_{20} \tag{6.23}$$

6.2 Heuristics for Truck Routes

Up to now, models that combine different pickup and delivery tasks to create connected components have been introduced; these models indirectly assign empty containers to cargo transportation requests. In the following, we denote by $\mathcal{W}$ the set of connected components that is computed by these models. The aim of this section is to develop routing heuristics that assign the set of connected components $\mathcal{W}$ to truck routes and determine a sequence in which the connected components are visited during a truck's route.

6.2.1 Preliminary Considerations

The first question to be addressed is how to define the problem that is to be solved by the routing heuristics. For $k \in \{20, 40\}$, Figure 6.3 depicts a representation of two connected components $w = (l_1, \ldots, l_n), w' = (l_m, \ldots, l_z) \in \mathcal{W}$ as sequences of pickup and delivery tasks $l_1, \ldots, l_n, l_m, \ldots, l_z \in \mathcal{E}_k \cup \mathcal{D}_k \cup \mathcal{F}_k$. Ideally, we would like to only consider such representations of connected com-

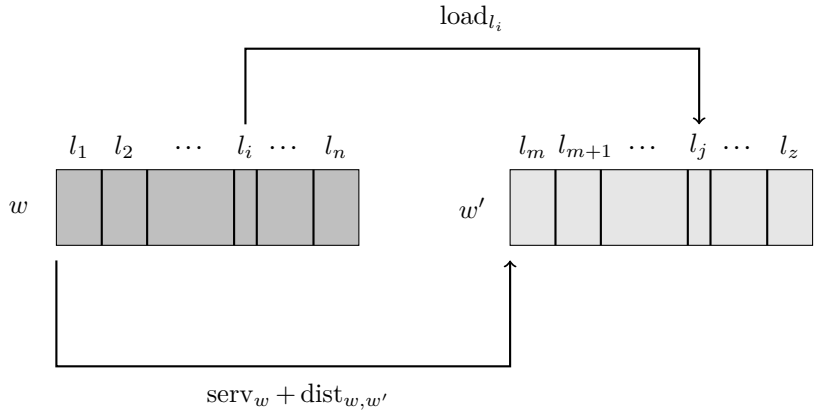


Figure 6.3: Representation of connected components as multi-task units.

ponents and no longer be aware of different pickup and delivery tasks; this representation would decrease the size of instances as well as the complexity of the problem definition considerably. In the following, we explain why it is not sufficient to only consider connected components.

At first glance, the problem seems to be an amTSPTW: the connected components are well-defined and built in such a way that the capacity of trucks is observed. The subsequent examples show why this is not the case. Figure 6.4 depicts two different connected components $w, w' \in \mathcal{W}$. Both connected

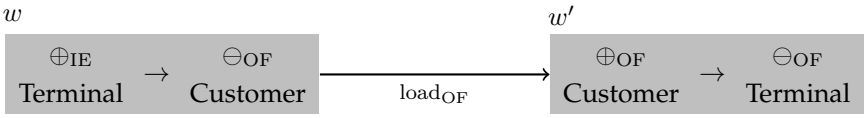


Figure 6.4: Precedence constraints between transportation requests.

components include tasks of the same OF request: the connected component w comprises a street-turn from an empty container pickup task of an IE request to the flexible empty container delivery task taking place at the customer's location of an OF request. The connected component w' comprises the subsequent well-defined task of the OF request, i.e. the transportation of the same container from the customer's location to the terminal. As the fully loaded container transportation is allowed to start only after the container has been loaded, the two connected components have to be synchronized when constructing routes. The additional requirement of observing *precedence constraints* (i.e. (un-)loading operations) between connected components, which share tasks of same OF/IF requests, results in a problem that is a generalization of the mTSPTWP (see also Section 3.1.2). For the construction of the set of precedence constraints an auxiliary set A_L is defined according to Definition 5.1; the set A_L contains ordered pairs (l_i, l_j), such that l_i is a delivery and l_j is a pickup task of the same OF/IF request taking place at the same customer's location, i.e. $\mathrm{loc}_i = \mathrm{loc}_j$. With the help of set A_L, a further set $A_L^{\mathcal{W}} \subset \mathcal{W} \times \mathcal{W}$ can be defined as follows:

$$A_L^{\mathcal{W}} := \{\, (w, w') \in \mathcal{W} \times \mathcal{W} \,|\, (l, l') \in A_L : l \in w, l' \in w', w \neq w' \,\} \tag{6.24}$$

Pairs of connected components $(w, w') \in A_L^{\mathcal{W}}$ share a precedence constraint, i.e. connected component w has to be completed a given amount of time before connected component w' is allowed to start. The problem definition in this section differs slightly from the definition of the mTSPTWP as we again (refer e.g. to Constraints 5.26) have to deal with variable precedence times resulting from the different possibilities for containers to be (un-)loaded; this is explained in more detail in the following. If the connected components $w, w' \in \mathcal{W}$ that are

shown in Figure 6.4 are assigned one after another to the same route of a truck, then the container is loaded in a stay-with procedure. Since the container remains on the truck while it is loaded in this case, the decoupling operation of the container from the truck (the last task of w) as well as the coupling operation of the same container to the truck (the first task of w') do not have to be executed. If w and w' are not assigned one after another to the same route of a truck, then the corresponding (de-)coupling operations must take place as the container is loaded in a drop-and-pick procedure. The construction of routes can lead to omitting durations for (de-)coupling only in case the first and last delivery and pickup tasks of connected components are contained as a pair in the set A_L. In this case, it is part of the routing heuristics to decide whether containers are (un-)loaded in a stay-with or a drop-and-pick procedure.

Unfortunately, the consideration of precedence constraints means that we cannot determine sizes of connected components before the entire set of truck routes is established; this fact holds true even if no time windows are considered, which is explained in the following. Figure 6.5 depicts two rows that represent routes of two different trucks. Six pickup and delivery tasks $l_1, \ldots, l_6$ are shown by six rectangles, which are displayed in four different colors. These different colors represent the underlying hinterland requests and depot container stowage operations of the mICTP instance. That is, the pickup task l_1 displays that one empty container is coupled by the first truck at the depot. In addition to the container stowage operation, the mICTP instance contains three different hinterland requests: two OE requests introduce two empty container delivery tasks l_2, l_3; one IF request introduces two fully loaded container pickup and delivery tasks l_4, l_5 and one empty container pickup task l_6. The tasks of the IF request are divided by precedence constraints, i.e. $(l_5, l_6) \in A_L$. Each pickup and delivery task is given a time window represented by the appropriated colored brackets in the row below the rectangle[1]. The tasks are assembled to connected components $w_1 = (l_1, l_2, l_6, l_3), w_2 = (l_4, l_5) \in \mathcal{W}$ that are depicted by dashed lined rectangles; the route of each truck includes exactly one connected component and it holds $(w_2, w_1) \in A_L^{\mathcal{W}}$. While the duration of connected component w_2 is as short as possible (tasks l_4 and l_5 are allowed to start immediately one after another), this does not apply to the duration of w_1, which includes some waiting time between l_2 and l_6. The delayed duration is achieved as l_6 is not allowed to start earlier because of the precedence time that has to be uphold between l_5 and l_6. Additionally, l_2 is not allowed to start later

[1]Be aware that by assumption the second visit of a customer is allowed to start at any point in time within the time horizon. This the reason for the time window of l_6 being larger than the time window of l_5, even though both tasks take place at the same location. The time window of l_1 represents the depot; this means that the time horizon reaches from zero to twelve.

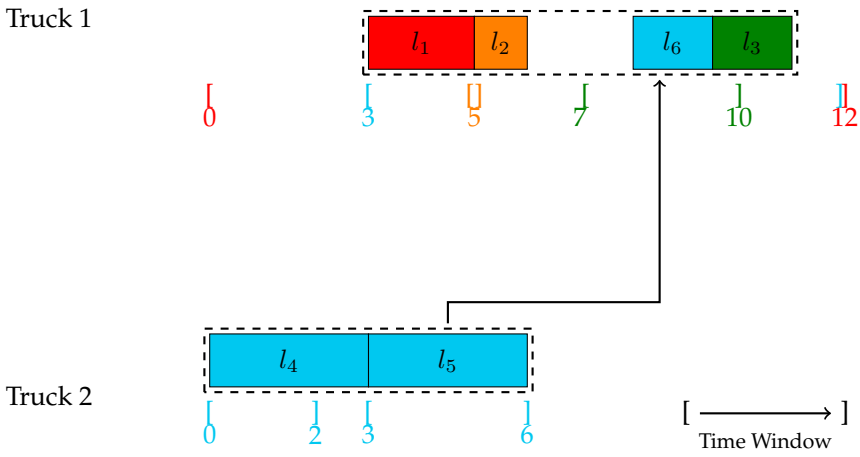


Figure 6.5: Varying durations of connected components.

because of its time window. This means that the duration of w_1 is dependent of the assignment of w_2 to a truck's route.

All in all, routing heuristics have to assign and sequence set $\mathcal{W}$ to the set of trucks $\mathcal{T}$. Each truck has to start and end its route at the depot d within the time horizon H; precedence constraints between the connected components, time windows of the different tasks and variable durations of the connected components have to be observed. The obtained problem belongs to the class of VRPMS; the interdependence of routes (refer to Definition 3.6) requires a fast method to check the feasibility of solutions.

6.2.2 Construction of Routes

Because of the considerations of the former section, the routing heuristics have to deal with a hybrid instance consisting of connected components on the one hand and pickup and delivery tasks on the other hand. Altogether, three different routing heuristics that are particularly introduced in Sections 6.3.1 and 6.3.2 are developed; all routing heuristics have a basic structure in common:

INSERTION HEURISTIC APPROACH

Input: A set of connected components $\mathcal{W}$ and a set of trucks $\mathcal{T}$.
Output: A set of feasible routes $\mathcal{R}$.

1. $\mathcal{R} \leftarrow \emptyset$
2. **while** $\mathcal{W} \neq \emptyset$ **do**
3. $t_{best} \leftarrow \emptyset$, cost $\leftarrow \infty$, $w \leftarrow$ choose($\mathcal{W}$)
4. **for all** $t \in \mathcal{T}$ **do**

5. **if** gap($\mathcal{R}, t$) < estimated_duration(w) **then**
6. **goto** 4
7. **for all** position($\mathcal{R}, t$) $\in t$ **do**
8. **if not**(feasible(position($\mathcal{R}, t$), w, $\mathcal{T}$)) **then**
9. **goto** 7
10. **if** cost > cost(position($\mathcal{R}, t$), w, $\mathcal{R}$, t) **then**
11. cost $\leftarrow$ cost(position($\mathcal{R}, t$), w,$\mathcal{R}$, t)
12. $t_{best} \leftarrow t$
13. **if** $t_{best} \neq \emptyset$ **then**
14. $\mathcal{R} \leftarrow$insert_best_position($\mathcal{R}, t_{best}, w$)
15. $\mathcal{W} \leftarrow \mathcal{W} \setminus \{w\}$
16. **else**
17. ... {proceed as explained in Sections 6.3.1 and 6.3.2}
18. **return** $\mathcal{R}$

The different routing heuristics iteratively choose elements that are contained in $\mathcal{W}$ (line 2) and try to insert the chosen element w in the best feasible position for insertion in a truck's route, whereby a set of evaluators that is introduced in Section 6.2.5 determines the quality of insertion positions. In this step, all positions (line 7) of all trucks (line 4) are considered. Lines 10 – 12 temporary store the best known insertion position and its cost. Lines 13 – 15 insert element w into the best known insertion position, in case a feasible insertion is possible. For a faster computation of solutions, line 5 considers an estimation of the duration of connected component w that is explained later by Formula 6.28; the estimated duration is compared with the difference between the time horizon H and the sum of durations of all connected components that are assigned to the currently tested route, i.e. the *gap* of the truck's route. If the estimated duration is larger than this gap, then the current iteration (truck) can be skipped for the search of an insertion position to w.

6.2.3 Feasibility Check

This section describes the feasibility check that is invoked in line 8 of the INSERTION HEURISTIC APPROACH. To check the feasibility of a currently made insertion or even an entire solution, all different pickup and delivery tasks (those tasks that are inserted into routes and those tasks that are not inserted into routes yet) are considered. The points of start t_i of the tasks l_i, the time horizon start t_s and the time horizon end t_e are used in order to match the different tasks in temporal relation. A *distance graph* is built out of the currently known constraints for the different points of start. Altogether four types of constraints can be distinguished at this point:

1. The earliest starting time of a pickup/delivery task l_i, i.e. the maximum of the time window start of l_i and the sum of the time window start of the depot d and the driving duration from d to l_i:

$$t_s + \max\{\underset{i}{\text{start}}, \underset{d}{\text{start}} + \underset{d,i}{\text{dist}}\} \leq t_i$$

2. The latest ending time of a pickup/delivery task l_i, i.e. the minimum from the time window end of l_i and the difference between the time window end of d (the horizon H) and the duration of l_i and the driving duration from l_i to d:

$$t_i + \min\{\underset{i}{\text{end}}, \underset{d}{\text{end}} - (\underset{i,d}{\text{dist}} + \underset{i}{\text{serv}})\} \leq t_e$$

3. The precedence time[2] that has to take place between two different pickup and delivery tasks $(l_i, l_j) \in A_L$:

$$t_i + \underset{i}{\text{load}} \leq t_j$$

4. The duration that has to take place between different pickup and delivery tasks l_i, l_j that are assigned one after another either to the same connected component or to the same route of a truck:

$$t_i + \underset{i,j}{\text{dist}} + \underset{i}{\text{serv}} \leq t_j$$

Each constraint is presented by an equation of type $x + a \leq y$, with $x, y \in \mathbb{R}$ are decision variables and $a \in \mathbb{R}$ is a constant. Dechter et al. [1991], Funke et al. [2016], Goel and Meisel [2013], Moffitt and Pollack [2006], Shostak [1981] show how these types of constraints can be easily transformed to a distance graph $G = (V, A)$. Each decision variable t_i, t_s, t_e introduces a node to the node set V. This means that the starting times of the depot are duplicated as they serve as the points of origin t_s and destination t_e of a truck's route; the remaining nodes are the representatives of the starting times t_i of the different pickup and delivery tasks. Each constraint $x + a \leq y$ introduces an arc (v_y, v_x) with cost $-a$ to the arc set A, whereby $v_x, v_y \in V$ denote the representative nodes of decision variables x, y. This step reverses the original order of the constraint's variables and negates the constant. The resulting graph G can be used to check the feasibility of a solution in polynomial time.

[2]For pickup and delivery tasks, which are assigned to truck routes or connected components, it is clear, whether (un-)loading operations take place in a stay-with or a drop-and-pick procedure; hence, this constraint can also consider the additional decoupling time that is needed by drop-and-pick procedure.

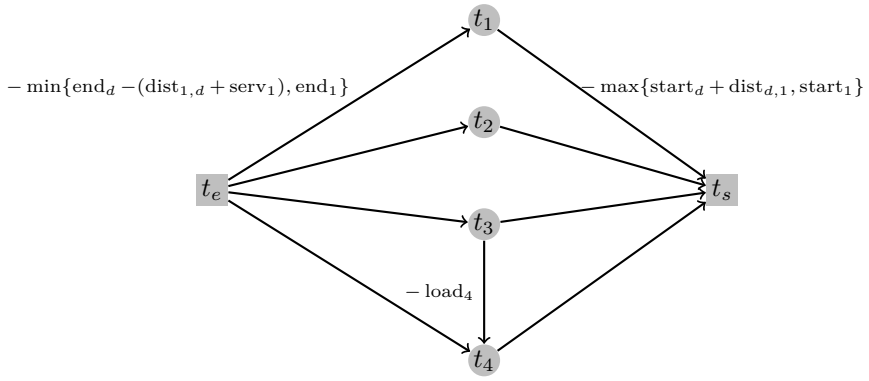


Figure 6.6: Distance graph (Funke and Kopfer [2015]).

By this definition, the absolute value of the length of a directed shortest path from t_e to t_s gives a lower bound on the operating time of a truck. If this value exceeds the time horizon, then the feasibility check fails. If the graph contains a negative directed cycle, then the feasibility check also fails since the existence of a negative directed cycle implies that there are constraints conflicting each other (Shostak [1981]). The feasibility check can be computed in polynomial time by solving the APSP in graph G (refer e.g. to Remark 2.26). As all arcs have assigned negative costs, the feasibility check can be implemented in $\mathcal{O}(|V|^2)$ (refer e.g. to Funke et al. [2016] for a proof). However, the present implementation of the feasibility check uses the algorithm of Floyd [1962], Warshall [1962] and therefore solves the APSP in $\mathcal{O}(|V|^3)$ time. The reason for choosing the algorithm of Floyd [1962], Warshall [1962] is that this algorithm constructs a distance matrix. Dechter et al. [1991] show how to extract a schedule for the pickup and delivery tasks from the rows of a distance matrix. Figure 6.6 depicts an example of a distance graph containing four pickup and delivery tasks l_1, l_2, l_3, l_4 with $(l_4, l_3) \in A_L$. For the sake of clarity, not all costs are written to arcs.

6.2.4 Estimations of Time Windows and Durations

The estimations of the connected components' time windows and durations remain to be investigated. The estimations of the connected components' durations are for example needed by line 5 of the INSERTION HEURISTIC APPROACH. The pairs of delivery and pickup tasks (l_i, l_j) that are contained in the set A_L once again require a special treatment. On the one hand, the associated (de-)coupling operations do not need to be implemented when l_i and l_j are allocated directly after each other in a connected component, since in

this case the container is (un-)loaded in a stay-with procedure. On the other hand, the duration of a connected component increases by the duration of the (un-)loading operation that has to take place between the starting times of l_i and l_j, if both tasks are included in the same connected component - no matter whether l_i and l_j are proceeded immediately one after another (stay-with procedure) or other tasks are proceeded in between (drop-and-pick procedure). However, the decision about what loading procedure is to be applied is made by the functions that assign containers to flexible tasks in advance.

In the following, the formulas for time windows and durations that are introduced by Zhang et al. [2010] are extended by the consideration of connected components comprising of more than two tasks and the precedences between the pairs of delivery and pickup tasks that are contained in the set A_L. For two delivery and pickup tasks l_i, l_j with $(l_i, l_j) \in A_L$, let $p(j) := l_i$ denote the predecessor of l_j and let $s(i) := l_j$ denote the successor of l_i. Let $p(i)/s(i) = \emptyset$, if l_i has no predecessor/successor. In order to include the choice of (un-)loading procedures that is taken by the routing heuristics in the display of the formulas, for each pair $(l_i, l_j) \in A_L$ a decision variable $\delta_{l_i,l_j} \in \{0, 1\}$ is introduced. The value of δ_{l_i,l_j} is set to one, if and only if l_i and l_j are allocated directly after each other in a truck's route. We set $\widetilde{\text{start}}_1 := \text{start}_1$, $\widetilde{\text{end}}_1 := \text{end}_1$, $\widetilde{\text{serv}}_1 := (1 - \delta_{p(l_1),l_1}) \cdot \text{serv}_1 + \text{dist}_{1,2}$ and $\text{dist}_{n,n+1} := 0$ and recursively define for $i \in \{2, \ldots, n\}$:

$$\widetilde{\text{start}}_i := \min\{\max\{\widetilde{\text{start}}_{i-1}, \underset{i}{\text{start}} - \widetilde{\text{serv}}_{i-1}\}, \widetilde{\text{end}}_{i-1}\} \tag{6.25}$$

$$\widetilde{\text{end}}_i := \min\{\widetilde{\text{end}}_{i-1}, \underset{i}{\text{end}} - \widetilde{\text{serv}}_{i-1}\} \tag{6.26}$$

$$\widetilde{\text{prec}}_i := \begin{cases} \text{load}_j + \widetilde{\text{serv}}_{j-1} - \text{dist}_{j,j+1}, & \text{if } (l_j, l_i) \in A_L, 1 \leq j < i \\ 0, & \text{otherwise} \end{cases} \tag{6.27}$$

$$\begin{aligned} \widetilde{\text{serv}}_i := \max\{&\underset{i}{\text{start}} - \widetilde{\text{end}}_{i-1}, \widetilde{\text{serv}}_{i-1}, \widetilde{\text{prec}}_i\} \\ &+ \underset{i,i+1}{\text{dist}} + (1 - \max\{\delta_{p(l_i),l_i}, \delta_{l_i,s(l_i)}\}) \cdot \underset{i}{\text{serv}} \end{aligned} \tag{6.28}$$

Having defined these formulas, then $\text{serv}_w := \widetilde{\text{serv}}_n$ is chosen for the estimation of the duration and $[\text{start}_w, \text{end}_w] := [\widetilde{\text{start}}_n, \widetilde{\text{end}}_n]$ is chosen for the estimation of the time window of connected component w. In addition, for two elements $w, w' \in \mathcal{W} \cup \{d\}$ the function $\text{dist}_{w,w'}$ is needed by the subsequent sections. Let the placeholder "-" represent an element of $\mathcal{W} \cup \{d\}$; if w is the depot ($w = d$), then the function $\text{dist}_{w,-}$ / $\text{dist}_{-,w}$ defines the distance from/to the depot. If w is a connected component ($w := (l_1, \ldots, l_n) \in \mathcal{W}$), then the function $\text{dist}_{w,-}$ / $\text{dist}_{-,w}$ is equal to $\text{dist}_{n,-}$ / $\text{dist}_{-,1}$.

6.2.5 Evaluators

Lines 10 – 12 and 14 of the INSERTION HEURISTIC APPROACH relate to the "best" feasible position for insertion of a connected component w into a route of a truck t. All in all, four different evaluators for the measurement of the quality of a solution are developed. Each call of the INSERTION HEURISTIC APPROACH arbitrarily selects one of the different evaluators. Let $w^-/w^+ \in \mathcal{W} \cup \{d\}$ denotes the predecessor/successor of the currently considered position in the route of truck t and let $[\text{start}_d, \text{end}_d] := [0, H]$ and $\text{serv}_d := 0$. We define the evaluators by considering the former introduced Formulas 6.25, 6.26 and 6.28:

- If the evaluator e_1 is chosen, then:

$$\text{cost(position}(\mathcal{R}, t), w, \mathcal{R}, t) := \underset{w^-,w}{\text{wait}} + \underset{w,w^+}{\text{wait}} - \underset{w^-,w^+}{\text{wait}}$$

  whereby for two elements $w, w' \in \mathcal{W} \cup \{d\}$ the value $\text{wait}_{w,w'}$ is defined according to $\text{wait}_{w,w'} := \max\{0, \text{end}_{w'} - \text{start}_w - \text{serv}_w - \text{dist}_{w,w'}\}$.

- If the evaluator e_2 is chosen, then:

$$\text{cost(position}(\mathcal{R}, t), w, \mathcal{R}, t) := \underset{w^-,w}{\text{dist}} + \underset{w,w^+}{\text{dist}} - \underset{w^-,w^+}{\text{dist}}$$

Evaluator e_3/e_4 diversifies the solution space by considering the maximum of zero and the sum of the objective of e_1/e_2 and the randomly chosen value *noise* $no \in \mathbb{R}$ (Ropke and Pisinger [2006], refer to Section 6.1.4).

6.3 Large Neighborhood Search

We follow the notations of Pisinger and Ropke [2010] to describe the entire heuristic approach. Functions including the term "containers" are based on the considerations of Section 6.1, while functions including the term "route" are based on the considerations of Section 6.2.

LARGE NEIGHBORHOOD SEARCH

Input: A problem instance $\mathcal{I}$.
Output: A feasible solution $\mathcal{S}$ of $\mathcal{I}$.

1. $\mathcal{S} \leftarrow$ assign_containers_only_empty($\mathcal{I}, \mathcal{S}$)
2. $\mathcal{S} \leftarrow$ route_subset_less_trucks($\mathcal{I}, \mathcal{S}$)
3. $\mathcal{S}^* \leftarrow \mathcal{S}, T \leftarrow T_0$
4. **repeat**
5. $\quad \mathcal{S}' \leftarrow \mathcal{S}$
6. $\quad$ **if** $\mathcal{S}'$ uses more trucks than specified **then**

7. $\mathcal{S}' \leftarrow$ improve_routes($\mathcal{S}', T$)
8. select destroy method $\delta \in \{\omega_1, \omega_2, \omega_3\}$,
   $\phi \in \{\text{rand}, \text{time}, \text{truck}, \text{worst\_travel}, \text{worst\_wait}\}$
9. $\mathcal{S}' \leftarrow$ remove_subset($\mathcal{I}, \mathcal{S}', \phi$)
10. **if** $\delta = \omega_2$ **then**
11. $\mathcal{S}' \leftarrow$ undo_container_assignment_only_empty($\mathcal{I}, \mathcal{S}'$)
12. $\mathcal{S}' \leftarrow$ assign_containers_only_empty($\mathcal{I}, \mathcal{S}'$)
13. **if** $\delta = \omega_3$ **then**
14. $\mathcal{S}' \leftarrow$ undo_container_assignment_full_and_empty($\mathcal{I}, \mathcal{S}'$)
15. $\mathcal{S}' \leftarrow$ assign_containers_full_and_empty($\mathcal{I}, \mathcal{S}'$)
16. select repair method $\rho \in \{\text{best\_insertion}, \text{greedy}, \text{regret}\}$
17. **if** $\rho = \text{best\_insertion}$ **then**
18. $\mathcal{S}' \leftarrow$ route_subset_best_insertion($\mathcal{I}, \mathcal{S}'$)
19. **if** $\rho = \text{greedy}$ **then**
20. $\mathcal{S}' \leftarrow$ route_subset_greedy($\mathcal{I}, \mathcal{S}'$)
21. **if** $\rho = \text{regret}$ **then**
22. $\mathcal{S}' \leftarrow$ route_subset_regret($\mathcal{I}, \mathcal{S}'$)
23. Accept $\mathcal{S}'$ with probability $p(T, \mathcal{S}', \mathcal{S})$
24. **if** cost($\mathcal{S}$) $<$ cost($\mathcal{S}^*$) **then**
25. $\mathcal{S}^* \leftarrow \mathcal{S}$
26. $\mathcal{S}' \leftarrow$ improve_routes($\mathcal{S}, T$)
27. $T \leftarrow T_{new}$
28. **until** Stop criterion is met
29. **return** $\mathcal{S}^*$.

An initial solution is built in lines 1 - 2 (refer to Section 6.3.1). Lines 4 - 28 try to improve the initial solution (refer to Section 6.3.2): three different destroy neighborhoods $\omega_1, \omega_2, \omega_3$ are invoked together with five different parameters $\text{rand}, \text{time}, \text{truck}, \text{worst\_travel}, \text{worst\_wait}$ (lines 8 – 15) and three different repair neighborhoods $\text{best\_insertion}, \text{greedy}, \text{regret}$ (16 – 22) add diversification to the heuristic approach. The function IMPROVE_ROUTES in lines 7 and 26 tries to improve truck routes without changing the current container assignment. The current solution is updated using SA in lines 3, 23 – 24 and 27 (refer to Section 6.3.3).

6.3.1 Construction of the Initial Solution

Up to now, the basic framework of the heuristic approach has been presented. This and the next section show how the constructive method and the subsequent local search operate in detail. The constructive method consists of two steps: first step involves the methods that are described in Section 6.1, line

1 constructs connected components by combining pickup and delivery tasks, second step involves the methods that are described in Section 6.2, line 2 assigns and sequences connected components onto truck routes by using a hierarchical objective, in which minimizing the number of used trucks is preferred to minimizing the total travel distance of trucks.

Assignment of Containers

In comparison to the local search algorithm, which only improves parts of an already existing solution, the constructive algorithm has to find a solution to an entire instance of the mICTP. Since exact approaches generate connected components, for a faster computation of an initial solution the instance is simplified. This means that the solution space of initial solutions is restricted to only contain the pickup and delivery tasks of empty containers, i.e. $\mathcal{F}_k$ is set equal to the empty set and the two Models $6.5 - 6.8$ and $6.9 - 6.23$ construct assignments of empty containers only. That is, the pickup and delivery tasks of fully loaded containers are excluded from the container assignment process of the constructive method. This simplification can be made as the transportation of fully loaded containers from their points of origin to their points of destination already constitutes a well-defined task. The constructive method's output $\mathcal{W}$ (the set of connected components) includes a set of connected components comprising the pickup and delivery tasks of empty containers that is joined with the set $\mathcal{W}_{\mathcal{F}}$; set $\mathcal{W}_{\mathcal{F}}$ denotes the set of connected components in which each fully loaded container transportation represents its own connected component (refer e.g. to the solid lined rectangles in Figure 6.1):

$$\mathcal{W}_{\mathcal{F}} := \bigcup_{r \in \mathrm{OF}_k \cup \mathrm{IF}_k, k \in \{20,40\}} (\oplus_r, \ominus_r) \tag{6.29}$$

As a consequence, the initial solution contains the direct movements of fully loaded containers between terminals and customers' locations or vice versa, only. Since trucks are able to simultaneously transport fully loaded and empty 20-foot containers, this simplification constitutes a restriction for the transportation of 20-foot containers, i.e. the blue, zigzag arcs in Figure 5.2 that form a special characteristic of the mICTP are banned from the solution space of the constructive method. As a result, it is a task of the local search algorithm to add the full variety of the different combination possibilities for empty and fully loaded 20-foot container transportation to the solution building process.

Insertion Heuristic Method

The function ROUTE_SUBSET_LESS_TRUCKS (line 2) is an adaption of an approach presented by Braekers et al. [2013], Jula et al. [2005]. At the beginning,

the approach computes a lower bound on the number of trucks that is sufficient to serve the set $\mathcal{W}$ of connected components (the construction of lower bounds is the topic of the next section). For the lower bound on the number of trucks, a routing heuristic approach is invoked $\chi \in \mathbb{N}$ times leading to the computation of χ different solutions. Afterwards, the number of trucks is incremented by one and the routing heuristic approach is invoked χ times again. The entire process is iterated until the number of trucks reaches the value $|\mathcal{T}|$. At the end, the initial solution is set to the best solution that has been computed by this approach.

The outline of the routing heuristic approach is given in Section 6.2.2. Line 3 of the INSERTION HEURISTIC APPROACH is instructed to choose one of the k best rated connected components $w \in \mathcal{W}$. The rating of a connected component is a lexicographic minimization of the following assets:

1. Connected components w sharing precedences with other connected components are preferred to others whenever all connected components w' with $(w', w) \in A_L^{\mathcal{W}}$ are served by trucks.

2. Connected components that are set to be *critical* are preferred to others.

3. Component w is preferred to w' when $\mathrm{serv}_w > \mathrm{serv}_{w'}$ (Formula 6.28).

4. Component w is preferred to w' when $\mathrm{start}_w < \mathrm{start}_{w'}$ (Formula 6.25).

5. Component w is preferred to w' when $\mathrm{end}_w < \mathrm{end}_{w'}$ (Formula 6.26).

If connected component w cannot be inserted into any of the routes in line 17 of the INSERTION HEURISTIC APPROACH, then an additional truck is added to $\mathcal{T}$ and w is tried to be inserted into the route of the new truck. If again w cannot be inserted into the route of the new truck, then the currently computed solution is rejected and w is set to be critical. Afterwards, the routing heuristic approach starts from scratch again. If the routing heuristic approach fails to compute routes containing all connected components for several iterations, then the function ASSIGN_CONTAINERS_ONLY_EMPTY is invoked in a tabu or noise-technique in order to compute a new set of connected components. A consequence of adding new trucks to $\mathcal{T}$ is that the initial solution might contain more than $|\mathcal{T}|$ routes. Consequently, the objective, i.e. the cost value cost($\mathcal{S}$) of a solution $\mathcal{S}$ in line 24 of the LNS, is as long multi-critical as the number of used trucks exceeds the number of specified trucks $|\mathcal{T}|$; the lexicographic order prefers minimizing the number of trucks to the actual objective that minimizes the total travel distance of trucks.

Computation of Lower Bounds on the Number of Trucks

The lower bound on the number of trucks is set to the maximum of the values lb_1, lb_2 that are constructed in dependency to the current container assignment that is stored in set $\mathcal{W}$. The value lb_1 is the result of the division of the connected components' total estimated duration (Formula 6.28) by the time horizon:

$$\underset{\mathcal{W}}{\text{serv}} := \sum_{w \in \mathcal{W}} (\underset{w}{\text{serv}} + \frac{\min_{w' \in \mathcal{W} \cup \{d\}} \{\text{dist}_{w',w}\} + \min_{w' \in \mathcal{W} \cup \{d\}} \{\text{dist}_{w,w'}\}}{2}) \quad (6.30)$$

$$lb_1 := \frac{\text{serv}_{\mathcal{W}}}{H} \quad (6.31)$$

The lower bound lb_1 is particularly effective for instances without time windows. Mitrović-Minić and Krishnamurti [2006] show that computing the minimum number of directed paths, which cover all nodes (in the following briefly called *Minimum Cover of Nodes by Directed Paths (MCNDP)*) in an *admissible graph* (i.e. a connected components' starting time precedence graph), gives a lower bound on the number of trucks of instances containing time windows. Value lb_2 is computed by this combinatoric approach. Therefore, an admissible graph $G = (V, A)$ is defined: each connected component induces one node in V; an arc is introduced into A whenever two connected components can be carried out one after another regarding the time windows:

$$\begin{aligned} V &:= \{\, v_w \mid w \in \mathcal{W} \,\}, \\ A &:= \left\{ (v_w, v_x) \in V \times V \;\middle|\; \underset{w}{\overset{'}{\text{start}}} + \underset{w,x}{\text{dist}} + \underset{w}{\text{serv}} \leq \underset{x}{\overset{'}{\text{end}}} \right\} \end{aligned} \quad (6.32)$$

In this formula serv_w describes the estimated duration (Formula 6.28) of connected components $w, x \in \mathcal{W}$ that are represented by nodes $v_w, v_x \in V$. The values of $\text{start}'_w, \text{end}'_x$ derive from the estimations of the starting and ending time of w, x (Formulas 6.25, 6.26) by additional including the depot that serves as the point of origin and destination of truck routes. This means that the start start'_w of connected component w is set to $\max\{\text{start}_w, \text{start}_d + \text{dist}_{d,w}\}$ and the end end'_x of connected component x is set to $\min\{\text{end}_x, \text{end}_d - (\text{dist}_{x,d} + \text{serv}_x)\}$.

Mitrović-Minić and Krishnamurti [2006] obtain a polynomial time algorithm for the construction of a lower bound from the proof of equivalence of Dilworth's theorem (refer to Theorem 2.31) and the theorem of König [1931] (refer to Theorem 2.14). The theorem of Dilworth stems from the field of order theory and considers *posets* (i.e. finite sets R together with a relation $\leq$ that is reflexive, antisymmetric and transitive, see also Definition 2.30). If a graph is acyclic and transitive, then a poset $(R, \leq)$ on a digraph $\widetilde{G} = (\widetilde{V}, \widetilde{A})$ can be defined by associating the set R with the set of nodes $\widetilde{V}$ and applying the relation $v \leq w$ to two elements $v, w \in R$ when $(v, w) \in \widetilde{A}$. The transitivity of $\widetilde{G}$

induces the transitivity of $(R, \leq)$ and since $\widetilde{G}$ is acyclic, $(R, \leq)$ is antisymmetric. In order to apply the theorem of Dilworth, it is necessary in a first step to transform the admissible graph G into an acyclic transitive graph $\widetilde{G} = (\widetilde{V}, \widetilde{A})$ (refer to Definitions 2.6 and 2.10). Mitrović-Minić and Krishnamurti [2006] consider the transitive closure (refer to Definition 2.10) of the condensation (refer to Definition 2.8) $\widetilde{G}$ of G instead of G. By definition, the transitive closure of a graph is a transitive graph and the condensation of a graph is an acyclic graph; both graphs can be computed in polynomial time[3] (refer to Remark 2.9 and Observation 2.27).

We are now able to apply the theorem of Dilworth to $\widetilde{G}$. That is, the minimum number of chains into which $(R, \leq)$ can be partitioned is equal to the MCNDP in $\widetilde{G}$, i.e. the minimum partition into chains gives a lower bound lb_2 on the number of trucks. The deduction of the theorem of Dilworth from the theorem of König shows a polynomial time transformation of $\widetilde{G}$ into a bipartite graph $\widetilde{G}'$ (refer to Definition 2.11). The MCNDP in $\widetilde{G}$ can be derived from a maximum bipartite matching in $\widetilde{G}' = (\widetilde{V}', \widetilde{A}')$. By Remark 2.18, the maximum bipartite matching problem can be solved in polynomial time. In order to provide the construction of $\widetilde{G}'$ that is also used by the present implementation, the deduction of Dilworth's theorem from König's theorem is stated:

Theorem 6.2 (Fulkerson [1956])**.** *König's theorem proves the theorem of Dilworth.*

Proof. We follow the proof that is shown in the book of Jacobs and Jungnickel [2004] and start with some notations that are needed to simplify the proof. For a graph G, let $\nu(G)$ denote the maximum cardinality of a matching in G and let $\tau(G)$ denote the minimum cardinality of a vertex cover in G (refer to Definition 2.12). For a poset $(R, \leq)$, let $\alpha(R, \leq)$ denote the maximum size of an antichain in $(R, \leq)$ and let $\Delta(R, \leq)$ denote the minimum number of chains into which $(R, \leq)$ can be partitioned.

Let $(R, \leq)$ be a poset and let n denote the cardinality of R. As an antichain contains at most one element of a chain, it holds

$$\alpha(R, \leq) \quad \leq \quad \Delta(R, \leq)$$

To proof the opposite direction, a partition of $(R, \leq)$ into $n - \nu(G)$ chains is constructed at first. Therefore a bipartite graph $G = (X \dot{\cup} Y, A)$ is defined. Each element $r \in R$ induces two copies x_r, y_r into the two bipartitions $x_r \in X$ and $y_r \in Y$. An arc is added to A when $r < r'$ holds for $r, r' \in R$, i.e.:

$$A := \{ (x_r, y_{r'}) \,|\, r, r' \in R, r < r' \}$$

[3] Mitrović-Minić and Krishnamurti [2006] state that the transitive closure can be computed in $\mathcal{O}(|V^c| \cdot |A^c|)$. However, as lower bounds are computed only once within the heuristic approach, in the present implementation the transitive closure is computed by the algorithm of Floyd [1962], Warshall [1962] that is also used by the feasibility check (refer to Section 6.2.3).

Let $M = \left\{ (x_r^i, y_{r'}^i) \,\middle|\, i \in [\nu(G)], r, r' \in R \right\}$ denote a maximum matching in G. Matching M is used for the construction of a partition C into chains in $(R, \leq)$. Since M is a matching, the elements $x_r^i, i \in [\nu(G)], r \in R$/the elements $y_{r'}^i, i \in [\nu(G)], r' \in R$ are pairwise disjoint. However, it is possible that M contains nodes corresponding to the copies of the same element $r \in R$ on both sides of the bipartition, i.e. x_r^i, y_r^j are both contained in M for some $i, j \in [\nu(G)], r \in R$. By uniting all pairs of chains $x_r^i < y_{r'}^i$ and $x_{r''}^j < y_r^j$ with x_r^i and y_r^j are the copies of the same element $r \in R$, a number $p \leq \nu(G)$ of distinct chains is obtained. Let c_i denote the size of chain $i \in [p]$. It holds $\nu(G) = \sum_{i=1}^{p}(c_i - 1)$. The remaining $n - (c_1 + \ldots + c_p)$ elements of R are grouped into trivial chains that consist of one element. The chains that are constructed from the matching together with the trivial chains form a partition into chains of $(R, \leq)$ that contains $n - (c_1 + \ldots + c_p) + p = n - \nu(G) \overset{\text{König}}{=} n - \tau(G)$ elements. Let VC denote a vertex cover in G. Vertex cover VC induces a set $W \subseteq R$ ($|W| \leq \tau(G)$) that contains at least one element of each chain $r < r'$, with $r, r' \in R$. This means that the complement $R \setminus W$ forms an antichain in $(R, \leq)$, which proves the opposite direction:

$$\alpha(R, \leq) \quad \geq \quad n - |W| \quad \geq \quad n - \tau(G) \quad \geq \quad \Delta(R, \leq)$$

□

The former proof states a construction of a transformation of $\widetilde{G}$ into a bipartite graph $\widetilde{G}' = (\widetilde{V} \times \widetilde{V}, \widetilde{E})$, with $\widetilde{E} := \left\{ \{v_w, v_x\} \,\middle|\, (v_w, v_x) \in \widetilde{A} \right\}$ that is also implemented in the present heuristic approach. Afterwards, the Maximum Flow algorithm of Ford and Fulkerson [1962] that computes in $\mathcal{O}(|\widetilde{V}' \times \widetilde{V}'| \cdot |\widetilde{E}'|)$ optimum solutions is implemented to obtain a maximum matching in $\widetilde{G}'$. Analogous to the former proof, the MCNDP in $\widetilde{G}$ can be deduced from a maximum matching in $\widetilde{G}'$ by uniting the pairs of the arcs of the matching that both correspond to the copies of the same element in $\widetilde{V}$ and inserting trivial paths for the elements of $\widetilde{V}$ that are not covered by any of the arcs of the matching.

6.3.2 Improvement of the Initial Solution

Up until now, an initial solution has been created. A local search algorithm tries to improve this initial solution. The individual steps of the local search algorithm proceed as follows. There are developed a variety of operators to choose a set of connected components from the initial solution. The chosen connected components are removed from the routes of the initial solution. Afterwards, one of many different operators inserts the connected components into newly computed positions. In this steps, the destroy as well as the repair

operators randomly choose one from the different evaluators that are defined in Section 6.2.5 in order to evaluate costs for removal and insertion.

Removal Heuristics

Line 8 of the LNS selects one of three different destroy methods $\omega_1, \omega_2, \omega_3$; each local search operator obtains one of five parameters $\mathrm{rand}, \mathrm{worst\_travel}, \mathrm{truck}, \mathrm{worst\_wait}, \mathrm{time}$ leading to twelve different strategies for the removal of connected components.

Let denote by $\mathcal{U} \subseteq \mathcal{W}$ the set of connected components that is being removed from truck routes. Five different parameters determine the choice of $\mathcal{U}$. Within the LNS, the parameters are randomly selected in line 8.

- If the parameter rand is chosen, then the connected components of the set $\mathcal{W}$ are randomly chosen and added to $\mathcal{U}$. The parameter rand tries to explore different areas of the solution space, while the other parameters try to identify and remove mistakes that were made by former iterations.

- If the parameter $\mathrm{worst\_travel}$ (Ropke and Pisinger [2006]) is chosen, then the connected components that result in the largest *savings* are inserted into $\mathcal{U}$. In this context, the saving of a connected component is defined as the difference between the total travel distance of trucks with and without the connected component being inserted into its current position. The parameter $\mathrm{worst\_travel}$ directly influences the actual objective.

- If the parameter $\mathrm{worst\_wait}$ is chosen, then the connected components that cause the largest waiting times are inserted into $\mathcal{U}$, whereby the waiting time between two connected components is defined analogously to the saving, but taking finishing time and starting time instead of travel distances into account. The parameter $\mathrm{worst\_wait}$ tries to shrink the number of used trucks by reducing the trucks' operating times.

- If the parameter truck is chosen, then a set of least occupied (i.e. the smallest number of served connected components) trucks is selected; connected components that are served by the selected trucks are inserted into $\mathcal{U}$. The parameter truck tries to reduce the number of used trucks.

- The parameter time is invoked together with a *reference time* (i.e. the current starting time of a randomly chosen connected component). The connected components with minimum deviation between their starting times and the reference time are inserted into $\mathcal{U}$. The parameter time selects components having assigned similar starting times within the current solution; the aim of this parameter is to compute a set of connected

components $\mathcal{U}$ that can easily exchange insertion positions, even though it is connected by a lot of constraints resulting from precedences.

After the set $\mathcal{U}$ has been formed, one of three neighborhoods $\omega_1, \omega_2, \omega_3$ is randomly selected in order to specify the degree of removal. Each neighborhood removes the set $\mathcal{U}$ from the routes of the current solution. The neighborhood ω_2 additionally invokes the function UNDO_CONTAINER_ASSIGNMENT-ONLY_EMPTY in line 11 of the LNS; this function defines that all connected components $w = (l_1, \ldots, l_n) \in \mathcal{U}$ that comprise only pickup and delivery tasks of empty containers, i.e. $l_1, \ldots, l_n \in \mathcal{D}_k \cup \mathcal{E}_k, k \in \{20, 40\}$, disperse. If the neighborhood ω_3 is selected, then the function UNDO_CONTAINER_ASSIGNMENT-FULL_AND_EMPTY in line 14 is invoked instead; this function defines all connected components breaking up in their individual pickup and delivery tasks.

Insertion Heuristics

Line 16 of the LNS selects one of three insertion heuristics $\mathrm{best\_insertion}, \mathrm{greedy}, \mathrm{regret}$ in order to reinsert $\mathcal{U}$ into truck routes again. However, in case one of the neighborhoods ω_2 and ω_3 has been invoked by the previous step, a new set of connected components has to be computed for all pickup and delivery tasks that are no longer contained in any connected component. The linear and integral programming techniques that are introduced in Section 6.1 are applied to construct connected components. In contrast to the constructive method that generates connected components comprising the entire set of the mICTP instance's pickup and delivery tasks, this step only affects the unassigned pickup and delivery tasks contained in $\mathcal{U}$. After the execution of neighborhood ω_2, the function ASSIGN_CONTAINERS_ONLY_EMPTY in line 12 of the LNS solves Model $6.5-6.8$ for empty 40-foot container assignment and Model $6.9-6.23$ for empty 20-foot container assignment, thereby it randomly choses, if the models are invoked normally or in a noise or a tabu technique (refer to Section 6.1.4). After the execution of neighborhood ω_2, the function ASSIGN_CONTAINERS-FULL_AND_EMPTY solves Model $6.5 - 6.8$ for empty 40-foot container assignment and Model 6.9 – 6.16 together with the adapted Constraints 6.18 – 6.21 for empty and fully loaded 20-foot container assignment. This step for the first time computes connected components comprising the pickup and delivery tasks of empty and fully loaded 20-foot containers (fully loaded container transportation is not considered by the constructive method)[4]. After the construction of new connected components, the feasibility is tested (Section 6.2.3).

[4] In the case of 40-foot containers (FTL problem) pickup and delivery tasks are still separately assigned, as this proceeding causes no restriction of the solution quality (Section 3.2).

If the feasibility check fails, then the current removal is reverted and a new iteration is initiated, i.e. the LNS proceeds from line 4.

If the feasibility check succeeds (neighborhoods ω_2, ω_3) or neighborhood ω_1 is invoked, then a set of connected components has to be inserted into the truck routes of the current solution. One of three different neighborhoods (routing heuristics) $\mathrm{best\_insertion}, \mathrm{greedy}, \mathrm{regret}$ is randomly selected. While the neighborhood $\mathrm{best\_insertion}$ is an adaptation of the best insertion method (Braekers et al. [2013], Jula et al. [2005]) that is also used by the constructive algorithm (refer to Section 6.3.1), the neighborhoods greedy and regret are also included in the ALNS of Ropke and Pisinger [2006]. Again, the outline of the routing heuristics is given in Section 6.2.2; the different routing heuristics differ primarily in the order, in which the connected components are chosen in line 3 of the INSERTION HEURISTIC APPROACH:

- The neighborhood $\mathrm{best\_insertion}$ randomly choses one of the k best rated connected components regarding the rating that is shown in Section 6.3.1.
- The neighborhood greedy choses the connected component that causes the least cost for insertion, i.e. the costs of all feasible insertion positions of all unassigned connected components have to be computed and compared.
- The neighborhood regret (also known as *Vogel's Approximation Method*, Reinfeld and Vogel [1958]) choses the connected component that obtains the maximum value of the difference between the costs of its best and its second best feasible insertion position, i.e. again the costs of all feasible insertion positions of all unassigned connected components have to be computed and compared. This neighborhood incorporates a look-ahead information into the heuristic approach.

If in any iteration a connected component is identified for which no feasible insertion position exists, then the current solution is rejected and a new iteration is initiated using the last known feasible solution, i.e. the LNS proceeds from line 4.

Improvement of Routes

The function IMPROVE_ROUTES (lines 7 and 26 of the LNS) uses a technique that is already known from the constructive algorithm (refer to Section 6.3.1). The routes are tried to be improved for a maximum of $\chi \in \mathbb{N}$ iterations without changing the current container assignment, i.e. a large number of connected components is removed by destroy method ω_1 that obtains one of five parameters $\mathrm{rand}, \mathrm{time}, \mathrm{truck}, \mathrm{worst\_travel}, \mathrm{worst\_wait}$. In each iteration, one of three

routing heuristics $\mathrm{best\_insertion}, \mathrm{greedy}, \mathrm{regret}$ is randomly chosen for the insertion of removed connected components. This process leads to the computation of χ different solutions; the solutions are updated in a SA technique that is explained in Section 6.3.3. The routes are tried to be improved at the beginning and the end of every iteration of the heuristic approach for the following reasons. Whenever the current solution contains more trucks than specified by the instance ($|\mathcal{T}|$), the function IMPROVE_ROUTES is invoked at the beginning of an iteration in line 7 of the LNS; the number of trucks is tried to be minimized by only allowing the parameter truck and evaluators e_1, e_3. Calling the function IMPROVE_ROUTES at the end of an iteration in line 26 of the LNS offers the possibility to diversify the solution space; different routing heuristics construct routes out of the high-quality container assignments that are computed by exact approaches.

Phases

The local search algorithm is divided into different phases that are controlled by four parameters $\theta_1, \theta_2, \theta_3, \theta_4 \in \mathbb{N}$. The parameter θ_1 determines the number of different phases. Let $\sigma \in \mathbb{N}$ denote the maximum number of iterations, i.e. if the local search algorithm reaches σ iterations, then the stop criterion (line 28 of the LNS) is met. One phase of the local search algorithm is defined to contain $\lfloor \frac{\sigma}{\theta_1} \rfloor$ iterations (remaining iterations are contained in the last phase). The value of the parameter η that controls the amount of noise, the number of connected components that are removed by removal heuristics and the number of trucks that are removed by parameter truck are decreased whenever a new phase is reached. Moreover, the tabu list that stores the arcs of the AP, MCFP and MFP graphs G_{20}, G_{40} is emptied. The different phases are introduced in order to intensify the search process. The parameters are changed according to the just mentioned proceeding for all but the last θ_2 phases. In the last θ_2 phases the algorithm is adjusted to search globally for solutions in the solution space, i.e. compared to the former phases, the number of connected components that are removed by removal heuristics and the number of trucks that are removed by parameter truck are increased in the last θ_2 phases.

Parameters θ_3 and θ_4 define a further stop criterion (line 28 of the LNS). If the local search algorithm has not been able to compute a better solution within the previous θ_3 iterations, then the number of connected components that are removed by removal heuristics and the number of trucks that are removed by parameter truck are increased. If the local search algorithm has not been able to compute a better solution within the previous θ_3 iterations for θ_4 times, then

it is aborted independently from the maximum number of iterations σ and the best known solution is returned.

6.3.3 Simulated Annealing

Within the local search algorithm current solutions are updated in a SA technique (refer to Section 2.2.3). In line 3 of the LNS some starting temperature $T \in \mathbb{R}$ is initialized. Let $\mathcal{S}$ denote the current solution and let $\mathcal{S}'$ denote the solution that has been recently computed by an iteration of the local search algorithm. For a solution $\mathcal{S}$ the value $\mathrm{cost}(\mathcal{S})$ is set to the total travel distance of trucks. If the term $\exp(-\frac{\mathrm{cost}(\mathcal{S}')-\mathrm{cost}(\mathcal{S})}{T})$ obtains a larger value than a randomly chosen number $r \in]0,1[$, then solution $\mathcal{S}'$ replaces solution $\mathcal{S}$. In each iteration of the local search algorithm the value of T is decreased by some factor that is determined in Section 7.4.1.

6.4 Implementation Details

The implementation of the heuristic approach invokes a commercial MIP solver for solving the Linear and Integer Programs 6.5 – 6.8, 6.9 – 6.16 and the adapted Constraints 6.18 – 6.21. Since these steps may demand a very large computation time, a fast implementation of the remaining algorithm is important. This section mentions some of the implementation details that are introduced in order to accelerate the computation time of the heuristic algorithm.

6.4.1 Partitioning of Large Instances

The constructive algorithm is a hybrid approach. The step of exactly assigning the entire set $\mathcal{E}_{20} \cup \mathcal{E}_{40}$ of empty containers' pickup and delivery tasks (Section 6.3.1, Formula 6.1) might consume a significant amount of computation time for large-sized instances. Therefore, the constructive algorithm partitions the instance into smaller subsets whenever the number of the instance's hinterland requests $|\mathcal{R}|$ exceeds a certain value $cr \in \mathbb{N}$ according to:

$$|\mathcal{R}| > 2 \cdot cr \qquad \text{or} \qquad |\,\mathrm{OF}\,| + |\,\mathrm{IF}\,| > cr \tag{6.33}$$

In order to partition the instance, the elements $l_i \in \mathcal{E}_{20} \cup \mathcal{E}_{40}$ are sorted in ascending order of their time windows' beginning start_i. The set $\mathcal{E}_{20} \cup \mathcal{E}_{40}$ is processed in the obtained order; mathematical models, i.e. the AP 6.5 – 6.8 and the MCFP 6.9 – 6.23, are separately invoked for sub-sets containing up to cr pickup and delivery tasks. Pickup and delivery tasks are stored in an additional list whenever the subsequent feasibility check proofs that the assignment

of these pickup and delivery tasks is infeasible. After the entire set $\mathcal{E}_{20} \cup \mathcal{E}_{40}$ has been processed, the mathematical models are invoked for the additional list in a noise or tabu technique (Section 6.1.4).

The partitioning approach affects the assignment of empty containers only, the subsequent construction of routes (Section 6.3.1) as well as the local search algorithm (Section 6.3.2) consider the entire instance again.

6.4.2 Implementation of the Feasibility Check

The feasibility check (Section 6.2.3) is invoked after each change (container assignment and route construction) of the solution. As a result, the major part of the total computation time of the heuristic approach is spend in the several executions of the feasibility check. In order to accelerate the computation time of the implementation of the feasibility check, an approach to decrease the number of pickup and delivery tasks, which serve as input for the feasibility check, is presented in the following. It follows from the solution's destruction taking place after its feasibility has been successfully checked that:

Observation 6.3. Since the function $\mathrm{dist}(-,-)$ satisfies the triangle inequality (refer to Section 4.1.2), the remaining routes of a solution remain feasible after the solution has been destroyed in lines 8 – 15 of the LNS.

Observation 6.3 implies that it is not necessary to introduce a representative node of every pickup and delivery task of the problem's instance to the distance graph that is build to check the feasibility of a repaired solution in lines 16 – 22 of the LNS. However, since the problem definition has to deal with the interdependence problem (refer to Definition 3.6), it is not sufficient to insert nodes representing the removed pickup and delivery tasks of set $\mathcal{U}$ only. Consequently, a set of *affected components* $\widetilde{\mathcal{U}}$ is recursively defined; set $\widetilde{\mathcal{U}}$ contains set $\mathcal{U}$ and all connected components that are assigned to routes, which might be affected by changes made by repair methods. For a truck $t \in \mathcal{T}$ let denote by $t := (w_1, w_2, \ldots, w_n)$ the route of t, i.e. the set of connected components $(w_1, w_2, \ldots, w_n) \subseteq \mathcal{W}$ that are assigned to t. The definition of set $\widetilde{\mathcal{U}}$ is given by:

---

1. $\widetilde{\mathcal{U}} \leftarrow \mathcal{U} \cup \{\, t \in \mathcal{T} \mid w \in (\mathcal{U} \cap t) \,\}$
2. **while** $\exists (w, w') \in A_L^{\mathcal{W}}$ or $\exists (w', w) \in A_L^{\mathcal{W}}$: $w \notin \widetilde{\mathcal{U}}$ and $w' \in \widetilde{\mathcal{U}}$ **do**
3. $\quad \widetilde{\mathcal{U}} \leftarrow \widetilde{\mathcal{U}} \cup \{\, t \in \mathcal{T} \mid w \in t \,\}$

Because of set $\widetilde{\mathcal{U}}$ being the minimum set of connected components to which the interdependence problem might apply, a feasibility check of the assignments of $\widetilde{\mathcal{U}}$ is sufficient to check the feasibility of the entire solution. The heuristic

approach's total computation time is enormously decreased be reducing the distance graphs' sizes in the feasibility check.

6.4.3 Route Assignment

The INSERTION HEURISTIC APPROACH, i.e. the three different insertion heuristics $\mathrm{best\_insertion}, \mathrm{greedy}$ and regret (Section 6.3.2), considers all insertion positions (line 7) of all truck routes (line 4) for the entire set of connected components $\mathcal{U}$. Each insertion position requires a feasibility check and a cost computation. Consequently, the computation time strongly increases with the size of the mICTP instance. For this reason, a further parameter $\zeta \leq |\mathcal{T}|, \zeta \in \mathbb{N}$ that restricts the number of truck routes for insertion is introduced. This means that if the size of a mICTP instance becomes too large, then the computation time of the heuristic algorithm is decreased by considering only the ζ less occupied trucks in line 4 of the INSERTION HEURISTIC APPROACH. Furthermore, for reasons of symmetry (homogeneous trucks' fleet) only one empty truck has to be considered for the insertion of a single connected component.

6.4.4 Computation Time of the Exact Approach

In order to avoid long computation times for solving an AP, MCFP or MFP instance (Models 6.5 – 6.8, 6.9 – 6.16 and 6.18 – 6.21) in the local search algorithm, the time limit of the commercial solver is set to ten seconds. If the commercial solver is not able to compute the optimum solution within ten seconds, then it returns the best available solution that has been computed within ten seconds. If the commercial solver is not able to compute any feasible solution, then the current step of the local search is aborted.

Chapter 7

Computational Study

The efficiency of the exact approach (Chapter 5) and the heuristic approach (Chapter 6) is evaluated by a computational study that involves randomly generated instances and benchmark instances known from literature sources. Section 7.1 introduces the different benchmark instances and some necessary adjustments that were made to implement the newly considered problem definition, i.e. the mICTP. Section 7.2 gives an overview of the subsequent computational study. Section 7.3 analyses the computational results of the exact algorithm. Section 7.4 primarily analyses the computational results of the heuristic algorithm, but also compares the computational results of both algorithms. Section 7.5 sums up the findings.

7.1 Test Instances

The computational study involves three different benchmark sets in total. The first two benchmark sets contain small-sized instances. These benchmark sets are generated by literature sources in order to demonstrate the complexity of the problem definition; each benchmark set is used by the corresponding literature source for the evaluation of their proposed mathematical model's efficiency. Funke and Kopfer [2016] randomly generate instances for the mICTP including six hinterland requests the most; Sterzik and Kopfer [2013] generate instances for the ICT including exactly eleven hinterland requests. In comparison to instances that can be optimally solved, a fleet of 30 trucks is able to handle up to 75 containers within one day in real-world applications (Srour et al. [2010], Wang and Regan [2002], Zhang et al. [2015]). Zhang et al. [2010] propose large-sized instances for the ICT that include 75 hinterland requests. The instances of Zhang et al. [2010] are also used by the computational studies of Nossack and Pesch [2013], Sterzik et al. [2012]. The ICT benchmark set

of Zhang et al. [2010] is adjusted to the mICTP in Section 7.1.2. Other sizes of instances used by literature sources reach from 100 (Jula et al. [2005]), 200 (Braekers et al. [2013], Caris and Janssens [2009], Imai et al. [2007]), 275 – 329 (Reinhardt et al. [2012]), 300 (Daham et al. [2016]), up to 500 hinterland requests (Chung et al. [2007], Zhang et al. [2015]).

7.1.1 Randomly Generated Instances

Funke and Kopfer [2016] randomly generate four benchmark sets in which the instances differ in the combination of hinterland request types, as shown in Table 7.1. Afterwards, the four sets of benchmark instances are duplicated; the time for (de-)coupling a container is set to five minutes in one duplication and to 20 minutes in the other duplication leading to eight different benchmark sets in total.

Set	$\lvert \mathrm{OF} \cup \mathrm{IF} \rvert$	$\lvert \mathrm{OE} \cup \mathrm{IE} \rvert$
(0, 6)	0	6
(1, 2)	1	2
(1, 3)	1	3
(2, 2)	2	2

Table 7.1: Layout of instances with up to six hinterland requests.

The locations of depots, terminals and customers are randomly located in the plane $[0, 10] \times [0, 10]$. The distance between two locations is set to the euclidean distance, which is multiplied by 1000 and cut before the first decimal in order to satisfy the triangle inequality; two locations are not more than four hours away from each other. The duration of (un-)loading a container is set to a random number varying between 20 minutes and two hours. Container sizes are randomly assigned to hinterland requests. In contrast to hinterland request types, container sizes, (un-)loading durations and locations, which do not change within a benchmark set, time windows (none, three different instance types with realistic time windows, three different instance types with randomly chosen time windows) and the number of trucks (1, 2, 3) vary. Consequently, a single benchmark set contains 21 different instances resulting in a total of 168 benchmark instances. The time horizon reaches from 0 a.m. to 24 p.m.; senders and receivers have limited opening hours with peak times between 6.00 a.m. to 9.00 a.m. and 5.00 p.m. to 8.00 p.m. for instance types having assigned realistic time windows (Van Der Horst and De Langen [2008]).

7.1.2 Instances of Literature Sources

Some adjustments are necessary in order to transform the ICT benchmark sets (Sterzik and Kopfer [2013], Zhang et al. [2010]) to instances for the mICTP.

ICT Instances of Sterzik and Kopfer [2013]

Sterzik and Kopfer [2013] generate a benchmark set for the ICT that contains ten instances (DS1, ..., DS10) on the basis of the geographical data and the time windows of customers, terminals and depots from the RC1-VRPTW-data sets of Solomon [1987]. Each instance includes eleven hinterland requests (five OF, five IF and one IE hinterland request), two depots, one terminal and five trucks. The following describes how to adapt the instances to obtain an adjusted instance set for the mICTP. Since the mICTP defines a single depot, the two depots are removed from the instances of Sterzik and Kopfer [2013] and a new depot is introduced and located at the position that is originally defined by Solomon [1987]. In order to obtain integral values, Sterzik and Kopfer [2013] round the euclidean distances; this procedure leads to distances that do not satisfy the triangle inequality in some cases. Distances that do not satisfy the triangle inequality are modified so that they satisfy the triangle inequality. The initial durations of (de-)coupling and (un-)loading containers are zero (Sterzik and Kopfer [2013]). The durations of (de-)coupling are set to two minutes and the durations of (un-)loading are set to ten minutes in the adjusted instance set. As a result, the time windows of the terminals of OF and IF hinterland requests are not only shifted by the driving durations between customers and terminals (this is the case for the instances of Sterzik and Kopfer [2013]), but also by the times that are needed to handle containers. The number of trucks is enlarged to ten in each instance[1].

Set	$\lvert \mathrm{OF}_{40} \cup \mathrm{IF}_{40} \rvert$	$\lvert \mathrm{OF}_{20} \cup \mathrm{IF}_{20} \cup \mathrm{IE}_{20} \rvert$
(0, 11)	0	11
(3, 8)	3	8
(5, 6)	5	6
(6, 5)	6	5
(8, 3)	8	3
(11, 0)	11	0

Table 7.2: Layout of instances with eleven hinterland requests.

[1] A similar benchmark set containing five to six trucks is considered by Funke and Kopfer [2016]. In order to analyze the solution quality of the heuristic algorithm, the number of trucks is increased to ten in the benchmark set of this thesis resulting in a larger solution space and more degrees of freedom.

The numbers of OF_{40} / OF_{20} and IF_{40} / IF_{20} requests vary in the instances, while geographical data, (un-)loading and (de-)coupling durations remain the same. Six different benchmark sets that are shown in Table 7.2 are obtained; each benchmark set comprises the ten instances of Sterzik and Kopfer [2013] leading to 60 instances in total.

ICT Instances of Zhang et al. [2010]

The benchmark set of Zhang et al. [2010] is comprised of 20 instances (Instance04, ..., Instance23). Each instance includes five depots, three terminals and 75 hinterland requests (40 IF, 30 OF and five IE hinterland requests). The initial times of the first time windows of hinterland requests are selected uniformly in the range $[0\,\mathrm{min}, 240\,\mathrm{min}]$; the initial times of second time windows are generated in dependency to the corresponding first time window and the point of start of the time horizon. The time windows' widths are randomly chosen in the interval $[1\,\mathrm{h}, 4\,\mathrm{h}]$. The (de-)couple durations of containers are set to five minutes and (un-)loading durations last between five and 60 minutes. Depots, terminals and customers are randomly located in the plane $[0\,\mathrm{min}, 180\,\mathrm{min}] \times [0\,\mathrm{min}, 180\,\mathrm{min}]$. Euclidean distances between locations are rounded in a way that complies with the triangle inequality in the adjusted benchmark set. Figure 7.1 depicts "Instance04". Again the original depots are removed from the instances and a single depot is newly introduced and located in the arithmetic mean of the locations of the five original depots. As a result, the time windows of the different locations have to be adapted. Let denote by d the new depot and let denote by $\mathcal{D}$ the set of original depots; for a hinterland request t denotes the corresponding terminal and c the corresponding customer. The first and second time windows of IF/IE hinterland requests are shifted by the value $\mathrm{dist}_{td} - \min_{d_i \in \mathcal{D}}\{\mathrm{dist}_{td_i}\}$; the two time windows of OF hinterland requests are shifted by the value $\mathrm{dist}_{td} - \min_{d_i \in \mathcal{D}}\{\mathrm{dist}_{td_i}\} + \mathrm{dist}_{cd} - \min_{d_i \in \mathcal{D}}\{\mathrm{dist}_{cd_i}\}$. While in the benchmark of Zhang et al. [2010] the number of trucks ranges from 50 to 56, each instance of the adjusted benchmark set contains exactly 56 trucks.

Once more, seven different benchmark sets are obtained by varying the numbers of OF_{40} / OF_{20}, IF_{40} / IF_{20} and IE_{40} / IE_{20} hinterland requests, while the geographical data, the (un-)loading and the (de-)coupling durations of the instance sets remain the same. The different benchmark sets are shown in Table 7.3. Each benchmark set contains the twenty instances of Zhang et al. [2010] leading to 140 instances in total.

Sections 7.4.1 and 7.4.2 consider a further benchmark set, which contains the twenty adjusted (geographical data, (un-)loading and (de-)coupling dura-

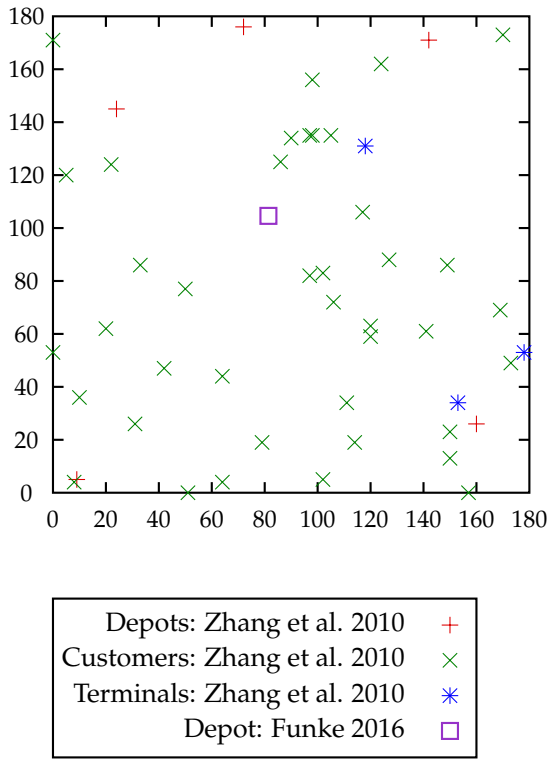


Figure 7.1: Instance 04, Zhang et al. [2010].

Set	$\lvert \mathrm{OF}_{40} \cup \mathrm{IF}_{40} \cup \mathrm{IE}_{40} \rvert$	$\lvert \mathrm{OF}_{20} \cup \mathrm{IF}_{20} \cup \mathrm{IE}_{20} \rvert$
(0, 75)	0	75
(12, 63)	12	63
(25, 50)	25	50
(37, 38)	37	38
(50, 25)	50	25
(63, 12)	63	12
(75, 0)	75	0

Table 7.3: Layout of instances with 75 hinterland requests.

tions) instances of Zhang et al. [2010] in which container sizes are randomly assigned to the different requests. As a result, the numbers of 20-foot and 40-foot hinterland requests differ from each other in this benchmark set. The term "mixed" refers to this benchmark set.

7.2 Structure of the Analysis

The exact algorithm and the heuristic algorithm are implemented in C++; mathematical models are implemented in C++ using the commercial solver IBM ILOG CPLEX Studio version 12.5.1. The following sections provide an analysis of the computational results of these implementations, which are produced on an Intel Core i5-3230, 2.6 GHz machine. The computational study is two-fold: on the one hand, Section 7.3 analyzes the solutions of the exact algorithm to small-sized instances that are defined in Section 7.1.1. On the other hand, the results of the heuristic algorithm are evaluated in Section 7.4; the instances containing up to eleven hinterland requests (Section 7.1.2) are used to compare the results of both algorithms, the instances containing up to 75 hinterland requests (Section 7.1.2) are used to show the applicability of the heuristic algorithm to real-world scenarios. The results are presented in the form of various tables in each of the following sections. As the tables have the same structure, the classification of columns is defined here as follows.

Each column contains either absolute values (marked by "Σ") or average values (marked by "ø"). Absolute Values ("Σ") refer to all benchmark instances that are listed in a row, whereby

- "Opt" is the number of instances that are solved to optimality,
- "Feas" is the number of instances that are solved without a proof of optimality, i.e. only an optimality gap is known for these instances,
- "Inf" is the number of instances for which the considered algorithm is unable to find a solution, when meeting its stop criterion,
- "Time" is the total computation time in seconds that the considered algorithm requires to solve all instances of the benchmark set.

Average Values ("ø") refer to those instances to which a feasible solution was computed, i.e. the instances that are listed in the columns "Opt" and "Feas", while instances contained in the column "Inf" are excluded. In these columns

- "Km" is the average value of the total travel distance of trucks,
- "Serv" is the average value of the total operating time of trucks,

- "$|\mathcal{T}|$" is the average value of the total number of trucks needed to serve the entire set of hinterland requests.

The column "Gap" contains an average value in percentage. The meaning of the column "Gap" differs depending on the considered algorithm. If the results of the exact algorithm are evaluated, then the column "Gap" refers to the average value on the maximum gap between the optimum solutions and the computed solutions. If the results of the heuristic algorithm are evaluated, then the column "Gap" refers to the gap between the objective values computed by the exact algorithm, which need not be optimum solutions, and the objective values computed by the heuristic algorithm.

If the content of a column is obvious, then the column is omitted.

7.3 Exact Approach

This section contains a study on the computational results of the exact algorithm, i.e. the implementation of the mathematical model. The results are based on an evaluation that is presented by Funke and Kopfer [2016]. The implementation of the mathematical model is able to solve instances regarding two different objectives. Objective 5.27 minimizes the total travel distance and Objective 5.28 minimizes the total operating time of trucks. The computation time of the exact algorithm to solve one single instance is limited to one hour.

Set	Exact, Objective 5.27				Exact, Objective 5.28						
	Opt	Inf	ø Km	Σ Time	Opt	Feas	Inf	ø Serv	ø Km	ø Gap	Σ Time
5min, all	80	4	21716	16	77	3	4	29651	23201	19	21493
5min, (0, 6)	19	2	28714	6	16	3	2	38414	30053	79	16547
5min, (1, 2)	21	0	8484	2	21	0	0	13440	9292	0	23
5min, (1, 3)	21	0	24104	2	21	0	0	30982	24912	0	80
5min, (2, 2)	19	2	26705	5	19	0	2	37332	29829	0	4843
20min, all	76	8	21293	15	73	3	8	34812	22982	16	19397
20min, (0, 6)	18	3	28423	4	15	3	3	46438	30373	68	14538
20min, (1, 2)	21	0	8484	2	21	0	0	16954	9696	0	19
20min, (1, 3)	19	2	24104	2	19	0	2	36432	25591	0	78
20min, (2, 2)	18	3	26139	6	18	0	3	42309	28338	0	4762

Table 7.4: Solutions to instances with up to six hinterland requests, exact approach (Funke and Kopfer [2016]).

Table 7.4 summarizes[2] the results of the exact algorithm for the benchmark set containing up to six hinterland requests, which is defined in Section 7.1.1. The rows of the table are subdivided into two sections regarding the different durations for (de-)coupling containers (five and 20 minutes). Each section

[2]For a more detailed analysis of the computational study presented in this section see also Tables A.1 and A.2 in the Appendix A.

comprises five rows; the first row contains results for the entire benchmark set, the subsequent rows contain results for the four sub-sets, which differ in the combination of hinterland request types (refer to Table 7.1). The columns of the table are also subdivided into two sections. The first section comprises four columns containing the results for minimizing the total travel distance of trucks, while the second section comprises seven columns containing the results for minimizing the total operating time of trucks. Minimizing the total operating time indirectly influences the total travel time. Therefore, the column "Km" is stated for both objectives. However, minimizing the total travel distance has no impact on the operating time, i.e. optimum travel distance solutions might contain routes ranging from the time horizon start to the time horizon end even though the truck finishes its last service earlier. As a result, the column "Serv" is omitted in the first section.

The overall conclusion is that minimizing the total travel distance is much easier than minimizing the total operating time; whereas computing optimum solutions to 84 test instances takes about sixteen seconds regarding the objective that minimizes the total travel distance, minimizing the total operating time takes between five to six hours and at least the proof of optimality is missing for three instances (see also columns "Time" and "Feas"). In the first set of rows, i.e. instances having (de-)coupling durations of five minutes, four instances are infeasible. This value increases to eight in the second set of rows. This increase is due to the second set of rows arising from the first set by only enlarging (de-)coupling durations but not changing time horizon and time windows. The varying number of infeasible instances is also the reason for the fact that the objective value minimizing the total travel distance decreases for the benchmark set in which the (de-)coupling duration is set to 20 minutes: the objective value is an average value in which infeasible instances are excluded.

7.4 Heuristic Approach

The following provides a comprehensive study on the efficiency of the heuristic algorithm, i.e. the implementation of the LNS. Section 7.4.1 starts with a selection of appropriate parameter settings of the heuristic approach in order to find high-quality solutions within reasonable computation time. Two different parameter settings are recommended for the different sizes of the benchmark instances, which vary between eleven (small-size) to 75 (large-size) hinterland requests. Since the heuristic algorithm includes a large number of local search operators, Section 7.4.2 analyzes the operators' influence on the solution quality. Section 7.4.3 contrasts the exact approach and the heuristic approach regarding solutions to small-sized benchmark instances. The entire study is

rounded up with the heuristic algorithm solving instances arising in real-world applications in Section 7.4.4.

7.4.1 Parameter Adjustments

Parameter	Description	Value
Mathematical Models, Section 6.1.4		
η	Disruptive factor, noise	0.75
η divisor	Divisor of the disruptive factor, noise	1.5
ξ	Tabu tenure	30
Constructive Method, Sections 6.3.2 and 6.4.1		
cr	No. requests using hybrid approach	11
χ	No. iterations IMPROVE_ROUTES	5
Insertion and Removal Heuristics, Sections 6.3.1, 6.3.2 and 6.4.3		
$\|\mathcal{U}\|$, first phases	Max. no. components for removal	6
$\|\mathcal{U}\|$, last phases	Max. no. components for removal	15
Rem. trucks	Max. no. trucks for removal (Parameter truck)	3
k	No. best rated components (ROUTE_SUBSET_LESS_TRUCKS)	15
ζ	Max. no. trucks for insertion	15
Simulated Annealing, Section 6.3.3		
T_0	Initial temperature	0.8
Cooling rate	Divisor of the temperature	0.2
Stop criterion for the LNS, Section 6.3.2		
σ	No. iterations	40
θ_1	No. phases	10
θ_2	No. last large phases	2
θ_3	No. iterations no improvement	3
θ_4	Max. no. times no improvement	6

Table 7.5: Default parameter setting of the heuristic approach.

The selection of the values for the different parameters of the heuristic algorithm is still pending. Table 7.5 summarizes the parameters that can be modified. The column "Value" refers to the initial value of the corresponding parameter that is listed in the column "Parameter", i.e. the value that was used before making the subsequent study. The study focuses on the highlighted cells, since it is assumed that these cells contain the parameters that have the highest impact on both the computation time and the solution quality. In the following we denote an ordered vector of parameters by

$$(a, b, c, d, e, f, g, h) \in \mathbb{N} \times \mathbb{N} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}$$

whereby a/b is the number of connected components $|\mathcal{U}|$ that are removed in the first/last phases, c is the initial value for the grade of noise η, d is the divisor of η, e is the divisor representing the cooling schedule of the SA, f is the

tabu tenure, g is the number of best rated connected components the function ROUTE_SUBSET_LESS_TRUCKS randomly selects from and h is the number of trucks that are included into the search for the best insertion position.

The structure of the subsequent study is as follows. First of all, several different parameter settings are tested for the benchmark set containing the adjusted instances of Sterzik and Kopfer [2013] (see also Section 7.1.2). Afterwards, the number of iterations is analyzed. Finally, a parameter setting for large-sized instances (adjusted instances of Zhang et al. [2010], Section 7.1.2) is determined on the basis of the results of small-sized instances.

Parameter Adjustments to Small-sized Instances

Set	Opt	Feas	Inf	ø $\lvert\mathcal{T}\rvert$	ø Objective		Σ Time	
					Km	Gap	Sec	Gap
exact	4	55	1	6.63	626.458	7.66	80658.10	–
exact and best known	4	56	0	6.60	625.133	–	77085.10	–
(6, 15, 0.75, 1.5, 0.2, 30, 15, 15)	1	59	0	6.03	664.567	6.31	1879.92	-97.56
(3, 15, 0.75, 1.5, 0.2, 30, 15, 15)	1	59	0	5.95	661.000	5.74	1601.99	-97.92
(12, 15, 0.75, 1.5, 0.2, 30, 15, 15)	1	59	0	5.95	662.700	6.01	1465.01	-98.10
(6, 7, 0.75, 1.5, 0.2, 30, 15, 15)	1	59	0	6.03	667.667	6.80	1022.36	-98.67
(6, 30, 0.75, 1.5, 0.2, 30, 15, 15)	1	59	0	6.02	664.050	6.23	2349.68	-96.95
(6, 15, 0.375, 1.5, 0.2, 30, 15, 15)	1	59	0	6.03	664.600	6.31	1362.63	-98.23
(6, 15, 1.5, 1.5, 0.2, 30, 15, 15)	1	59	0	5.98	664.033	6.22	1940.86	-97.48
(6, 15, 0.75, 0.75, 0.2, 30, 15, 15)	1	59	0	5.98	660.550	5.67	1231.58	-98.40
(6, 15, 0.75, 3.0, 0.2, 30, 15, 15)	1	59	0	6.02	662.700	6.01	1641.45	-97.87
(6, 15, 0.75, 1.5, 0.1, 30, 15, 15)	1	59	0	6.02	666.750	6.66	1875.83	-97.57
(6, 15, 0.75, 1.5, 0.4, 30, 15, 15)	1	59	0	6.03	664.567	6.31	1874.13	-97.57
(6, 15, 0.75, 1.5, 0.2, 15, 15, 15)	1	59	0	6.03	664.567	6.31	1867.76	-97.58
(6, 15, 0.75, 1.5, 0.2, 60, 15, 15)	1	59	0	6.03	664.567	6.31	1842.13	-97.61
(6, 15, 0.75, 1.5, 0.2, 30, 7, 15)	1	59	0	5.92	665.750	6.50	1505.69	-98.05
(6, 15, 0.75, 1.5, 0.2, 30, 30, 15)	1	59	0	5.95	659.717	5.53	1539.89	-98.00
(6, 15, 0.75, 1.5, 0.2, 30, 15, 7)	2	58	0	6.00	665.767	6.56	1387.89	-98.20
(6, 15, 0.75, 1.5, 0.2, 30, 15, 30)	1	59	0	6.00	665.300	6.43	1441.51	-98.13

Table 7.6: Solutions to instances with eleven hinterland requests, different parameter settings of the heuristic approach.

The solutions to the adjusted instances of Sterzik and Kopfer [2013] are listed in Table 7.6; the results of the exact approach (first and second row) are compared with the results of the heuristic approach (subsequent rows), which receives different parameter vectors. Starting with the initial parameter vector "(6, 15, 0.75, 1.5, 0.2, 30, 15, 15)" in the third row, the parameter vectors of the subsequent rows are generated by halving and doubling exactly one entry. Since the instances in the newly considered benchmark set are almost twice as large as the instances in the benchmark set that is considered in Section 7.3, a

further stop criterion is added to the exact approach: the exact algorithm stops, if it proofs the currently best known solution to be at most 5% far away from the optimum solution. These settings allow the exact approach to compute solutions to all but one instance (refer to the column "Inf") within one hour. The instance DS8 (0, 11) is not solved by the exact approach. In comparison, the heuristic approach computes solutions to all instances. For this reason, the second row "exact and best known" is added to the table. The row "exact and best known" contains the 59 solutions that are computed by the exact approach together with the best known solution[3] to the instance DS8 (0, 11) that is found by the heuristic approach. Consequently, the columns "ø Objective – Gap" and "Σ Time – Gap" of the heuristic approach relate to the row "exact and best known" (bearing in mind that the computation time of the exact approach additionally includes the time to proof the solution's optimality, e.g. compliance with the 5% gap to optimality). The exact approach computes optimum solutions to the four instances DS10 (8, 3), DS4 (11, 0), DS5 (11, 0) and DS6 (11, 0) in a matter of seconds. These results indicate that it is easier for the exact approach to compute solutions to instances including more hinterland requests for 40-foot containers than hinterland requests for 20-foot containers. Within a few seconds the heuristic approach computes solutions to the entire benchmark set. The gap between the total objective value computed by the heuristic approach and the total best known objective value ranges from 5.53% to 6.80%.

The following analysis investigates the maximum number σ of the local search iterations of the heuristic approach. The analysis is based on computations in which the heuristic algorithm receives the parameter vector "(6, 15, 0.75, 1.5, 0.2, 30, 30, 15)" (the highlighted row in Table 7.6). Figure 7.2 shows the development of the objective value of the three instance sets DS2, DS3 and DS6 over 40 local search iterations[4]. Each instance set contains six instances in which geographical data and container handling durations are fixed; only the numbers of OF_{40} / OF_{20} and IF_{40} / IF_{20} requests vary within an instance set. Since a street-turn can only take place between outbound and inbound requests when the requests define the transportation and handling of containers sharing the same size, the varying numbers of request types in an instance set lead to instances (3, 8), (5, 6), (6, 5) and (8, 3) being not comparable to each other: although these instances are contained in the same instance set, the combination possibilities of hinterland requests by street-turns may differ. However,

[3] The calculated objective value of the best known solution to instance DS8 (0, 11) is 547. This value is computed in 25.94 seconds by the heuristic approach that receives the parameter vector "(6, 15, 0.75, 1.5, 0.2, 30, 7, 15)".

[4] If no improvement is found within a specific number of iterations, the algorithm terminates before reaching 40 iterations (see also Section 6.3.2 for an explanation of the different stop criteria).

solutions to the instances of the same instance set are at least as good as the solution quality of instance (0, 11). This is because the requests of instance (0, 11) consider 20-foot containers only and, therefore, this instance includes all combination possibilities of the other instances, i.e. the solution spaces of the other instances are contained in the solution space of instance (0, 11). The fact that the objective value of instance (0, 11) (red curve in Figure 7.2) does not always lie below the other curves demonstrates once more the complexity of computing solutions to instances considering many 20-foot container hinterland requests.

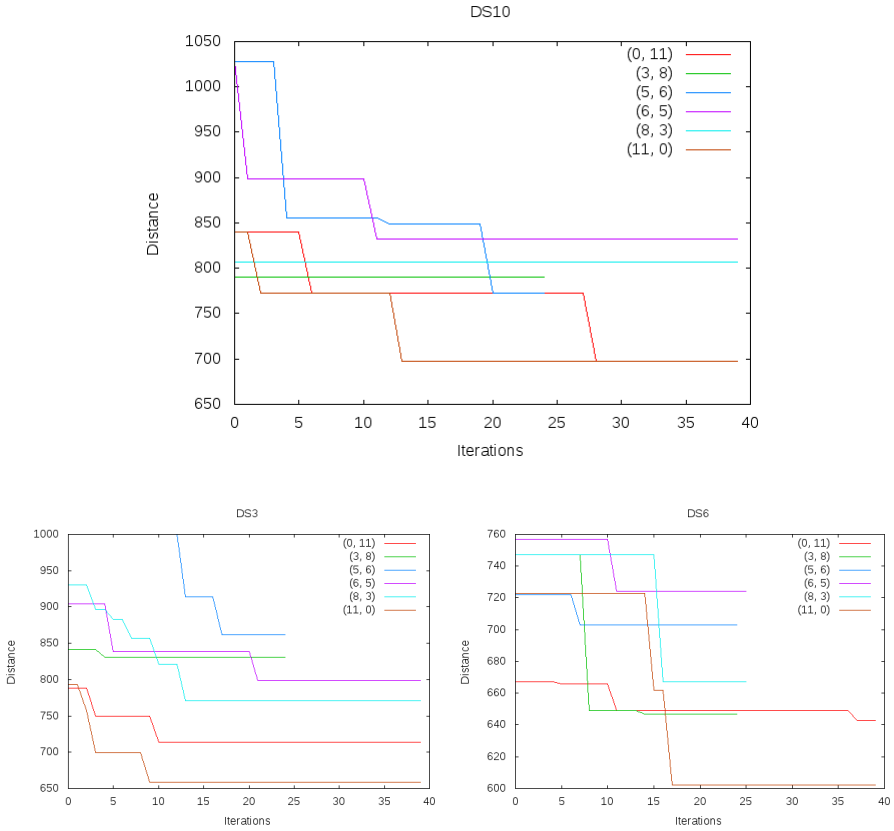


Figure 7.2: Development of the objective of instances with eleven hinterland requests.

The results of the different computations shown in this section are used to recommend a maximum number of local search iterations for the computation of solutions to large-sized instances. Since the computation of solutions to large-sized instances requires substantially more computation time than the computations in this section, the aim is to find high-quality solutions within a reasonable amount of time in case that larger instances are considered. Instance sets DS3 and DS6 are not improved by the last iterations; in contrast, for

instance DS10 (0, 11) the last iterations lead to great improvements. Altogether, most of the instances are mainly improved within 25 iterations. Therefore, the value of σ is set to 25 in the subsequent analysis.

Parameter Adjustments to Instances Inspired by the Real World

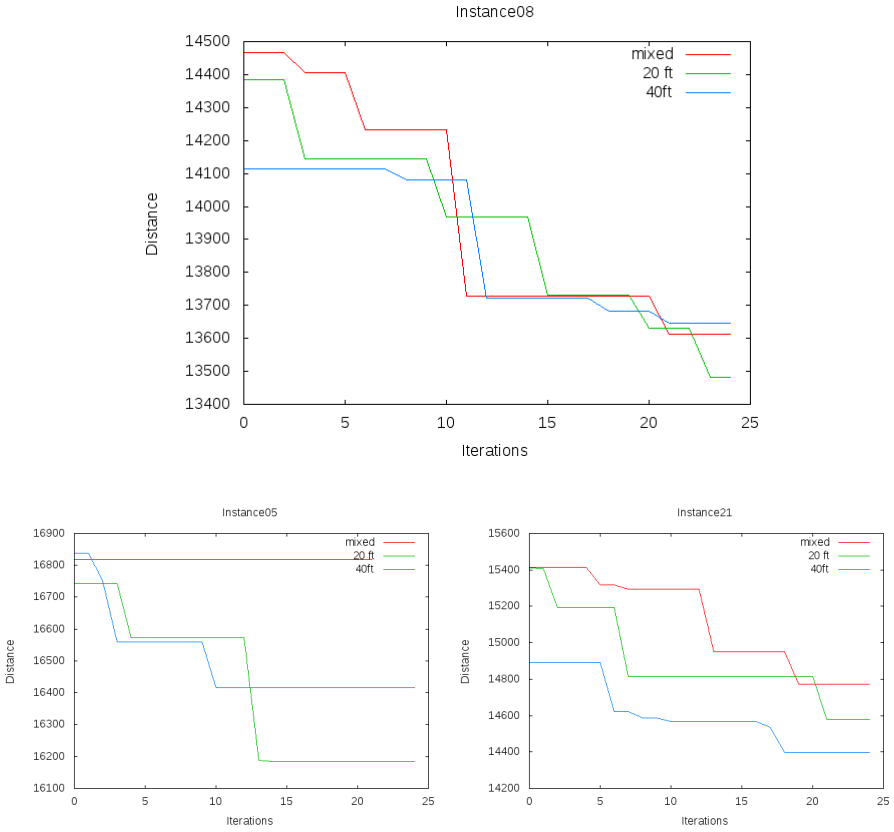


Figure 7.3: Development of the objective of instances with 75 hinterland requests.

The following analysis concerns a recommendation for the parameter set to be applied to large-sized instances. The subsequent study is based on the adjusted instances of Zhang et al. [2010]. Due to the long computation times, the study does not include all the container combinations that are listed in Table 7.3. Instead, the study covers 60 instances that are comprised of 20 instance sets, which arise from the 20 instances of Zhang et al. [2010] by tripling the original instance. The first and second instances of each instance set contain only 40-foot ("(75, 0)") or 20-foot containers ("(0, 75)") respectively, while the third instance contains a randomly chosen combination of both types of containers ("mixed") (see also Section 7.1.2). Because of the significant reduction of the value of σ to 25 iterations, some parameters need to be adapted before-

hand in order to balance the restricted number of iterations. This is explained in more detail in the following. Computations that comprise a large number of iterations are able to diversify the solution space. Therefore, these computations are allowed to comprise a great random factor as well, leading for instance to larger values for the penalization η that disturbs the objective value, and the number of best rated connected components k among which the function ROUTE_SUBSET_LESS_TRUCKS randomly chooses (see also Section 6.3.1). In contrast, if the value of σ decreases, the approach needs to be intensified, which results in higher computation times of one single iteration. The following study is based on computations in which the number of iterations χ that improve routes (see also Section 6.3.2) and the number of trucks that are removed by the parameter truck are doubled. The value of ζ is decreased in order to reduce the computation time of the search for solutions to large-sized instances, meaning that the search for the best insertion position no longer investigates all trucks. A more detailed definition of the different parameter values is stated in Table 7.8.

Two conclusions follow from the previous section. On the one hand, the values for kilometers only differ by around 1.2% between the worst and best parameter setting (667.667 to 659.717, refer to Table 7.6). On the other hand, solutions to instances containing many 40-foot containers are generally much better than solutions to instances containing many 20-foot containers (see Figure 7.2). Therefore, the choice of parameters for this analysis focuses on a different treatment for 40-foot and 20-foot containers. Let an ordered vector of parameters be denoted by

$$(a, b, c, d, e, f, g) \in \mathbb{N} \times \mathbb{N} \times \mathbb{R} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}$$

whereby a is the tabu tenure, b is the number of trucks that are included in the search for the best insertion position, c is the initial value for the grade of noise η, d/e is the number of connected components $|\mathcal{U}|$ containing 20-foot/40-foot containers only that are removed in the first phases, f/g is the number of connected components $|\mathcal{U}|$ containing 20-foot/40-foot containers only that are removed in the last phases. The different values (60, 7)/(30, 15) for the tuple (a, b) are combined with all combinations of the values: 0.1875, 0.375 for c, 6 and 12 for d and e, and 15 and 30 for f and g.

Table 7.7 shows the solutions to the adjusted instances of Zhang et al. [2010], which are computed by the heuristic approach that receives different parameter vectors. The best known average solutions are marked in green. Regarding the computation time, it is much better to set (a, b) to (60, 7) than to set (a, b) to (30, 15). For this reason, the heuristic approach receives the highlighted parameter vector "(60, 7, 0.375, 6, 6, 15, 30)" for the computation of solutions to large-

Set, (0, 75)	ø Km	Σ Time	Set, (75, 0)	ø Km	Σ Time
(30, 15, 0.1875, 12, 12, 15, 15)	15898.7	10063.30	(30, 15, 0.1875, 12, 12, 15, 15)	15619.1	14699.90
(30, 15, 0.1875, 12, 12, 30, 30)	15882.5	9739.36	(30, 15, 0.1875, 12, 12, 30, 30)	15577.5	15077.60
(30, 15, 0.1875, 6, 6, 15, 15)	15902.7	10568.40	(30, 15, 0.1875, 6, 6, 15, 15)	15683.7	14957.60
(30, 15, 0.1875, 6, 6, 30, 30)	15881.0	10444.10	(30, 15, 0.1875, 6, 6, 30, 30)	15661.6	15359.30
(30, 15, 0.375, 12, 12, 15, 15)	15930.2	9596.43	(30, 15, 0.375, 12, 12, 15, 15)	15561.6	14861.90
(30, 15, 0.375, 12, 12, 30, 30)	15908.9	10171.70	(30, 15, 0.375, 12, 12, 30, 30)	15570.0	14954.20
(30, 15, 0.375, 6, 6, 15, 15)	15859.4	8916.85	(30, 15, 0.375, 6, 6, 15, 15)	15620.4	14127.10
(30, 15, 0.375, 6, 6, 30, 30)	15834.0	9837.36	(30, 15, 0.375, 6, 6, 30, 30)	15618.4	14263.80
(60, 7, 0.1875, 12, 12, 15, 15)	15904.2	2656.05	(60, 7, 0.1875, 12, 12, 15, 15)	15800.5	3430.33
(60, 7, 0.1875, 12, 12, 30, 30)	15930.3	2922.20	(60, 7, 0.1875, 12, 12, 30, 30)	15821.6	3398.13
(60, 7, 0.1875, 6, 6, 15, 15)	15945.1	2675.90	(60, 7, 0.1875, 6, 6, 15, 15)	15800.4	3218.04
(60, 7, 0.1875, 6, 6, 30, 30)	15904.5	2870.70	(60, 7, 0.1875, 6, 6, 30, 30)	15786.2	3223.12
(60, 7, 0.375, 12, 12, 15, 15)	15937.3	2647.64	(60, 7, 0.375, 12, 12, 15, 15)	15725.4	3656.72
(60, 7, 0.375, 12, 12, 30, 30)	15949.0	2610.85	(60, 7, 0.375, 12, 12, 30, 30)	15702.0	3618.88
(60, 7, 0.375, 6, 6, 15, 15)	15781.4	3048.43	(60, 7, 0.375, 6, 6, 15, 15)	15730.0	3578.35
(60, 7, 0.375, 6, 6, 30, 30)	15741.9	3142.07	(60, 7, 0.375, 6, 6, 30, 30)	15753.1	3515.19

Set, mixed	ø Km	Σ Time	Set, mixed	ø Km	Σ Time
(30, 15, 0.1875, 12, 12, 15, 15)	15775.1	6041.57	(60, 7, 0.1875, 12, 12, 15, 15)	15798.8	2018.28
(30, 15, 0.1875, 12, 12, 15, 30)	15774.0	6610.94	(60, 7, 0.1875, 12, 12, 15, 30)	15813.3	1950.34
(30, 15, 0.1875, 12, 12, 30, 15)	15765.0	6120.66	(60, 7, 0.1875, 12, 12, 30, 15)	15818.1	2229.06
(30, 15, 0.1875, 12, 12, 30, 30)	15755.8	6608.52	(60, 7, 0.1875, 12, 12, 30, 30)	15813.9	2134.81
(30, 15, 0.1875, 12, 6, 15, 15)	15784.5	5753.77	(60, 7, 0.1875, 12, 6, 15, 15)	15835.7	1760.75
(30, 15, 0.1875, 12, 6, 15, 30)	15760.5	5897.21	(60, 7, 0.1875, 12, 6, 15, 30)	15805.1	1760.23
(30, 15, 0.1875, 12, 6, 30, 15)	15767.8	6033.52	(60, 7, 0.1875, 12, 6, 30, 15)	15848.4	1845.76
(30, 15, 0.1875, 12, 6, 30, 30)	15773.8	5879.51	(60, 7, 0.1875, 12, 6, 30, 30)	15807.5	1876.37
(30, 15, 0.1875, 6, 12, 15, 15)	15677.6	6632.09	(60, 7, 0.1875, 6, 12, 15, 15)	15794.5	1992.29
(30, 15, 0.1875, 6, 12, 15, 30)	15706.4	6554.83	(60, 7, 0.1875, 6, 12, 15, 30)	15794.5	1876.91
(30, 15, 0.1875, 6, 12, 30, 15)	15658.7	7013.55	(60, 7, 0.1875, 6, 12, 30, 15)	15765.0	2227.75
(30, 15, 0.1875, 6, 12, 30, 30)	15672.9	7076.34	(60, 7, 0.1875, 6, 12, 30, 30)	15748.0	2044.24
(30, 15, 0.1875, 6, 6, 15, 15)	15686.6	6846.04	(60, 7, 0.1875, 6, 6, 15, 15)	15783.6	1824.35
(30, 15, 0.1875, 6, 6, 15, 30)	15677.9	6181.30	(60, 7, 0.1875, 6, 6, 15, 30)	15749.5	1800.28
(30, 15, 0.1875, 6, 6, 30, 15)	15693.6	6623.44	(60, 7, 0.1875, 6, 6, 30, 15)	15775.4	2102.78
(30, 15, 0.1875, 6, 6, 30, 30)	15665.5	6619.11	(60, 7, 0.1875, 6, 6, 30, 30)	15757.0	1962.62
(30, 15, 0.375, 12, 12, 15, 15)	15820.9	5322.85	(60, 7, 0.375, 12, 12, 15, 15)	15827.1	1954.26
(30, 15, 0.375, 12, 12, 15, 30)	15826.6	5828.98	(60, 7, 0.375, 12, 12, 15, 30)	15825.0	1928.96
(30, 15, 0.375, 12, 12, 30, 15)	15810.9	5545.16	(60, 7, 0.375, 12, 12, 30, 15)	15839.5	2179.01
(30, 15, 0.375, 12, 12, 30, 30)	15805.3	5929.61	(60, 7, 0.375, 12, 12, 30, 30)	15811.8	2171.84
(30, 15, 0.375, 12, 6, 15, 15)	15783.9	5508.27	(60, 7, 0.375, 12, 6, 15, 15)	15871.8	1826.90
(30, 15, 0.375, 12, 6, 15, 30)	15774.3	5546.06	(60, 7, 0.375, 12, 6, 15, 30)	15854.0	1827.43
(30, 15, 0.375, 12, 6, 30, 15)	15766.0	5983.15	(60, 7, 0.375, 12, 6, 30, 15)	15866.6	1869.19
(30, 15, 0.375, 12, 6, 30, 30)	15784.2	5706.12	(60, 7, 0.375, 12, 6, 30, 30)	15868.4	1891.06
(30, 15, 0.375, 6, 12, 15, 15)	15674.5	6921.13	(60, 7, 0.375, 6, 12, 15, 15)	15834.9	1920.04
(30, 15, 0.375, 6, 12, 15, 30)	15708.1	6750.88	(60, 7, 0.375, 6, 12, 15, 30)	15834.8	1861.88
(30, 15, 0.375, 6, 12, 30, 15)	15677.4	6919.19	(60, 7, 0.375, 6, 12, 30, 15)	15852.3	1980.05
(30, 15, 0.375, 6, 12, 30, 30)	15677.5	6813.45	(60, 7, 0.375, 6, 12, 30, 30)	15847.2	2024.12
(30, 15, 0.375, 6, 6, 15, 15)	15672.0	6867.42	(60, 7, 0.375, 6, 6, 15, 15)	15829.8	1905.82
(30, 15, 0.375, 6, 6, 15, 30)	15686.2	6104.42	(60, 7, 0.375, 6, 6, 15, 30)	15810.8	1795.05
(30, 15, 0.375, 6, 6, 30, 15)	15665.2	6857.81	(60, 7, 0.375, 6, 6, 30, 15)	15836.9	2035.82
(30, 15, 0.375, 6, 6, 30, 30)	15650.2	6494.81	(60, 7, 0.375, 6, 6, 30, 30)	15817.4	2005.76

Table 7.7: Solutions to instances with 75 hinterland requests, different parameter settings of the heuristic approach.

sized instances in the remainder of this chapter. Figure 7.3 shows the development of the objective value for the instance sets Instance05, Instance08 and Instance21 over 25 local search iterations. This time, the development of the objective values of Instance05 and Instance08 behaves as expected; that is, 20-foot container transportation receives the best total travel distance. However, Instance21 shows that these promising results do not apply to all instances. Moreover, the best known solution to Instance05 (mixed) is computed by the constructive method.

Final Parameter Settings

Parameter	Description	Value	
		≤ 11	≈ 75
Mathematical Models, Section 6.1.4			
η	Disruptive factor, noise	0.75	0.375
η divisor	Divisor of the disruptive factor, noise	1.5	1.5
ξ	Tabu tenure	30	60
Constructive Method, Sections 6.3.2 and 6.4.1			
cr	No. requests using hybrid approach	11	11
χ	No. iterations IMPROVE_ROUTES	5	10
Insertion and Removal Heuristics, Sections 6.3.1, 6.3.2 and 6.4.3			
$\|\mathcal{U}\|$, first phases	Max. no. components for removal 20 ft	6	6
$\|\mathcal{U}\|$, first phases	Max. no. components for removal 40 ft	6	6
$\|\mathcal{U}\|$, last phases	Max. no. components for removal 20 ft	15	15
$\|\mathcal{U}\|$, last phases	Max. no. components for removal 40 ft	15	30
Rem. trucks	Max. no. trucks for removal (Parameter truck)	3	6
k	No. best rated components (ROUTE_SUBSET_LESS_TRUCKS)	30	15
ζ	Max. no. trucks for insertion	10	7
Simulated Annealing, Section 6.3.3			
T_0	Initial temperature	0.8	0.8
Cooling rate	Divisor of the temperature	0.2	0.2
Stop criterion for the LNS, Section 6.3.2			
σ	No. iterations	40	25
θ_1	No. phases	10	10
θ_2	No. last large phases	2	2
θ_3	No. iterations no improvement	3	3
θ_4	Max. no. times no improvement	6	6

Table 7.8: Final parameter setting of the heuristic approach.

Table 7.8 summarizes the recommended parameter values for the different numbers of hinterland requests. We recommend the parameter vector (6, 15, 0.75, 1.5, 0.2, 30, 30, 15) for instances containing less than or equal to eleven hinterland requests and we recommend the parameter vector (60, 7, 0.375, 6, 6, 15, 30) for instances containing around 75 hinterland requests.

7.4.2 Analysis of the Implemented Operators

The last test is carried out to analyze the influence on the solution quality and the computation time of the different variations of operators that are implemented in the heuristic approach. The computational study in this section uses the 60 adjusted instances of Zhang et al. [2010] that comprise the sets (0, 75), (75, 0) and mixed (see also Section 7.4.1 for further details regarding these instances).

Set	Feas	ø $\lvert\mathcal{T}\rvert$	ø Km	Σ Time
Original	60	43.57	15777.9	8403.83
Evaluators, Section 6.2.5				
Only e_2	60	43.52	15798.5	11647.10
Without e_3, e_4	60	43.38	15781.4	7961.33
Insertion Heuristics, Section 6.3.2				
Without best_insertion	60	43.53	15888.5	8796.85
Without greedy	60	43.50	15773.7	7875.14
Without regret	60	43.50	15796.5	7867.13
Removal Heuristics, Section 6.3.2				
Without rand	60	43.32	15795.2	8700.26
Without worst_travel	60	42.92	15813.6	9238.68
Without worst_wait	60	43.30	15731.4	9470.67
Without truck	60	43.78	15831.7	9104.16
Without time	60	43.35	15819.5	7808.31
Update Strategy, Section 6.3.3				
Hill climbing	60	43.57	15783.5	8569.50

Table 7.9: Results of the implemented operators for 75 hinterland requests.

Table 7.9 shows the results of the different implementations of the heuristic approach in which some of the operators are omitted. The first row contains the results of the implementation comprising the entire set of operators, which are introduced in Chapter 6, i.e. this is the implementation that is investigated before and after this section. The next two rows show the impact of the different evaluators measuring the quality of an insertion. The row "only e_2" contains the results of the heuristic approach, when only evaluator e_2 is used. This evaluator implements the actual objective and, therefore, the diversification of the solution space is severely restricted. The following row "Without e_3, e_4" contains results, which are obtained by omitting evaluators e_3 and e_4, i.e. diversification remains limited as no noise is investigated. Both rows have in common that the solution quality deteriorates, but the computation time improves for not implementing the noise technique. The following eight rows include results that are computed by omitting exactly one of the eight different insertion and removal operators. The overall results of the operators are positive as most of the times the objective value and the com-

putation time increase. However, the objective value improves when ignoring the operators greedy and worst_wait. Since the differences are marginal for the operator greedy, while the computation time significantly increases for the operator worst_wait, these operators remain implemented in the further analysis. Furthermore, the results of the operators best_insertion and truck seem to be very promising. The final row considers the influence of the update strategy; the SA technique is replaced by the hill climbing technique. The differences between the results of the two techniques are surprisingly small.

The noise and the tabu technique that are used to diversify the solution space of the container assignments (refer also to Section 6.1.4) are not considered in this section. This analysis is left out for two reasons. On the one hand, the influence of the parameters η, ξ that control the amount of noise and tabu penalization has already been investigated when adjusting these parameters in the former sections. On the other hand, omitting tabu or noise technique for larger instances may result in the constructive method not even finding an initial solution to larger instances.

7.4.3 Solution Quality

Table 7.10 compares the results of the heuristic approach that receives the parameter vector (6, 15, 0.75, 1.5, 0.2, 30, 30, 15) with the results of the exact approach for the adjusted benchmark set of Sterzik and Kopfer [2013] (see also Section 7.1.2). The exact approach minimizes the total travel distance; its computation time for solving one single instance is limited to one hour. Furthermore, the exact algorithm is again allowed to terminate, if it computes a solution that is at most 5% far away from the optimum solution.

Set	Exact						LNS		
	Opt	Feas	Inf	ø Km	ø Gap	Σ Time	ø Km	ø Gap	Σ Time
all	4	55	1	626.458	7.66	80658.10	659.717	–	1539.89
DS1	0	6	0	485.000	3.88	2457.70	512.000	5.57	172.35
DS2	0	6	0	639.333	8.88	10983.40	690.333	7.98	64.32
DS3	0	6	0	719.500	8.41	8682.15	748.000	3.96	145.81
DS4	1	5	0	620.500	9.86	7327.59	664.833	7.14	163.57
DS5	1	5	0	616.833	5.44	7205.89	645.500	4.65	186.25
DS6	1	5	0	606.500	8.95	7236.40	653.500	7.74	182.56
DS7	0	6	0	650.833	10.07	10874.50	664.333	2.07	47.78
DS8	0	5	1	606.800	7.40	10811.40	615.500	–	137.05
DS9	0	6	0	574.000	5.72	7216.89	614.667	7.08	342.27
DS10	1	5	0	742.000	7.97	7862.42	788.500	6.27	97.95

Table 7.10: Summary of solutions to instances with eleven hinterland requests.

As already mentioned, there is a difference between the column "ø Gap" of the exact approach and the heuristic approach. The column "ø Gap" of the

Set	Exact			LNS		
	ø Km	ø Gap	Σ Time	ø Km	ø Gap	Σ Time
DS2, all	639.333	8.88	10983.40	690.333	7.98	64.32
DS2, (0, 11)	559	26.74	3598.75	648	15.92	6.93
DS2, (3, 8)	560	7.80	3599.82	642	14.64	20.89
DS2, (5, 6)	674	6.05	3599.86	674	0.00	21.28
DS2, (6, 5)	727	5.00	184.29	826	13.62	8.95
DS2, (8, 3)	658	4.64	0.57	694	5.47	4.65
DS2, (11, 0)	658	3.04	0.03	658	0.00	1.61
DS7, all	650.833	10.07	10874.5	664.333	2.07	47.78
DS7, (0, 11)	570	25.47	3598.65	607	6.49	22.63
DS7, (3, 8)	668	15.22	3599.92	685	2.54	7.34
DS7, (5, 6)	712	7.05	3599.96	724	1.69	6.50
DS7, (6, 5)	710	5.00	75.57	712	0.28	3.85
DS7, (8, 3)	655	4.27	0.35	668	1.98	5.86
DS7, (11, 0)	590	3.39	0.02	590	0.00	1.60
DS8, all	606.800	7.40	10811.40	615.5	–	137.05
DS8, (0, 11)	–	–	3598.80	558	–	65.01
DS8, (3, 8)	621	17.60	3599.81	686	10.47	22.39
DS8, (5, 6)	579	6.34	3599.88	606	4.66	35.85
DS8, (6, 5)	603	5.00	12.56	606	0.50	6.84
DS8, (8, 3)	681	4.43	0.32	687	0.88	5.44
DS8, (11, 0)	550	3.64	0.01	550	0.00	1.51

Table 7.11: Detailed solutions to instance sets DS2, DS7 and DS8.

exact approach contains the average value of the maximum gaps between the solutions that are computed by the exact approach and the optimum solutions, while the column "ø Gap" of the heuristic approach contains the average value of the deviations between the solutions computed by the heuristic approach and the solutions computed by the exact approach. The deviations between the results of the heuristic approach and the results of the exact approach range from 2.07% to 7.98%. In contrast to the exact approach, the heuristic approach computes solutions to all instances. Within one hour, the exact algorithm is not able to compute a solution to instance DS8 (0, 11).

Table 7.11 shows detailed results for the three[5] instance sets DS2, DS7 and DS8. Regarding the objective values, the heuristic approach performs best for benchmark set DS2, while it performs worst for benchmark set DS7. Benchmark set DS8 contains the instance to which the exact approach does not compute any solution within one hour. A proof of optimality is not given to any solution that is mentioned in the table. The rows of the table are subdivided into three sections regarding the different benchmark sets DS2, DS7 and DS8. Each section is comprised of six rows; the first row ("DS2, all", "DS7, all" and "DS8, all") contains average values for the different benchmark sets, while the

[5]Further results for the single instances are listed in Table A.3 in the Appendix A.

subsequent rows contain results of the single instances. Once more it becomes obvious from the columns "ø Gap" and "Σ Time" that both algorithms perform much better when fewer 20-foot container hinterland requests are included in the input instances. In particular, the heuristic approach and the exact approach compute the same solutions to all instances in Table 7.11 that include 40-foot container hinterland requests only.

7.4.4 Application to Real-world Instances

A real-world application of the heuristic algorithm that receives parameter vector (60, 7, 0.375, 6, 6, 15, 30) is stated in the following. The study includes the adjusted benchmark set of Zhang et al. [2010] together with all combinations of hinterland requests that are shown in Table 7.3. Table 7.12 gives an overview of

Set	Feas	ø $\lvert\mathcal{T}\rvert$	ø Km	Σ Time
all	140	42.91	15802.3	17505.300
Instance04	7	46.00	15519.3	686.179
Instance05	7	46.14	16233.7	795.646
Instance06	7	47.43	14150.9	791.200
Instance07	7	49.86	15206.6	6326.000
Instance08	7	43.14	13688.6	703.669
Instance09	7	45.71	18509.6	623.056
Instance10	7	47.43	15437.9	765.485
Instance11	7	48.29	16725.3	682.326
Instance12	7	41.43	13636.3	705.419
Instance13	7	52.71	19626.3	595.452
Instance14	7	40.14	15595.6	1179.810
Instance15	7	43.57	16298.6	599.465
Instance16	7	45.14	15366.1	654.727
Instance17	7	43.57	16370.0	509.057
Instance18	7	44.00	14866.6	831.756
Instance19	7	39.29	17554.9	532.980
Instance20	7	29.57	14087.6	1673.370
Instance21	7	34.43	14638.6	1458.240
Instance22	7	36.14	17426.1	911.985
Instance23	7	34.14	15108.3	2173.470

Table 7.12: Summary of solutions to instances with 75 hinterland requests.

the computational results. The heuristic approach computes solutions to all instances. The average computation time to solve a single instance is 125.038 seconds. Since a commercial solver computes the solutions to the sub-problems, the computation time for instances of this size is very short. Table 7.13 provides further details of the computational results of benchmark sets Instance04, Instance07, Instance10, Instance13, Instance16 and Instance21. In the case of almost all instances, the best objective value (marked in green) is computed for either instances containing many 40-foot container hinterland requests or for instances containing many 20-foot container hinterland requests. This result

is not surprising, since rising the number of different combinations of 20-foot and 40-foot container hinterland requests in one single instance results in a prevention of street-turns between specific hinterland requests.

Once more, the best result within one instance set can be used as a lower bound on the minimum improvement of the instances "(0, 75)" that contain 20-foot container hinterland requests only. Deviations from this minimum improvement vary between 0% (Instance07, Instance16), 0.31% (Instance21), 1.97% (Instance13) and 2.15% (Instance04) up to 3.10% (Instance10). Further deviations that arise from Table A.4 in the Appendix A are 0% (Instance22), 0.79% (Instance08), 2.48% (Instance05), 2.58% (Instance17), 2.84% (Instance09), 3.46% (Instance15), 3.71% (Instance18), 4.16% (Instance06), 4.69% (Instance19) and 5.40% (Instance11), with a total of four outliers, 8.18% (Instance14), 13.04% (Instance20), 16.82% (Instance12) and 18.65% (Instance23).

Figure 7.4 shows the development of the objective value for instance sets[6] Instance05, Instance07, Instance10, Instance13, Instance16 and Instance21 over 25 iterations. Determining the number of iterations to 25 seems to be a reasonable choice. Benchmark sets Instance10, Instance13 and Instance21 yield very good solution qualities for instances containing 40-foot container hinterland requests only. However, sometimes the heuristic approach computes good solutions to instances containing many 20-foot containers, like instance sets Instance07 and Instance16.

[6]Further results that are not mentioned in this section are shown in Figures A.1 and A.2 in the Appendix A.

Set	ø $\lvert\mathcal{T}\rvert$	ø Km	Σ Time	Set	ø $\lvert\mathcal{T}\rvert$	ø Km	Σ Time
Instance04, all	46.00	15519.3	686.179	Instance13, all	52.71	19626.3	595.452
Instance04, (0, 75)	47	15535	87.616	Instance13, (0, 75)	54	19660	102.110
Instance04, (12, 63)	46	15856	84.307	Instance13, (12, 63)	52	19798	51.011
Instance04, (25, 50)	46	15562	54.910	Instance13, (25, 50)	54	19738	76.628
Instance04, (37, 38)	46	15238	77.188	Instance13, (37, 38)	54	19651	63.236
Instance04, (50, 25)	47	15732	77.492	Instance13, (50, 25)	54	19369	84.369
Instance04, (63, 12)	45	15504	149.613	Instance13, (63, 12)	50	19887	108.882
Instance04, (75, 0)	45	15208	155.053	Instance13, (75, 0)	51	19281	109.216
Instance07, all	49.86	15206.6	6326.000	Instance16, all	45.14	15366.1	654.727
Instance07, (0, 75)	49	14866	112.617	Instance16, (0, 75)	44	14607	220.637
Instance07, (12, 63)	50	15474	104.718	Instance16, (12, 63)	43	14997	73.652
Instance07, (25, 50)	51	15650	81.655	Instance16, (25, 50)	44	15256	69.478
Instance07, (37, 38)	51	15021	83.051	Instance16, (37, 38)	44	15311	145.329
Instance07, (50, 25)	51	15261	46.752	Instance16, (50, 25)	43	14972	79.782
Instance07, (63, 12)	49	15236	88.081	Instance16, (63, 12)	43	15619	21.573
Instance07, (75, 0)	48	14938	115.132	Instance16, (75, 0)	55	16801	44.277
Instance10, all	47.43	15437.9	765.485	Instance21, all	34.43	14638.6	1458.240
Instance10, (0, 75)	48	15108	126.937	Instance21, (0, 75)	34	14432	333.614
Instance10, (12, 63)	48	15811	105.832	Instance21, (12, 63)	34	14858	109.234
Instance10, (25, 50)	48	16005	84.803	Instance21, (25, 50)	35	14387	230.740
Instance10, (37, 38)	47	15769	115.263	Instance21, (37, 38)	33	14704	181.730
Instance10, (50, 25)	48	15328	101.739	Instance21, (50, 25)	34	14934	172.876
Instance10, (63, 12)	48	15390	85.606	Instance21, (63, 12)	36	14756	219.642
Instance10, (75, 0)	45	14654	145.305	Instance21, (75, 0)	35	14399	210.402

Table 7.13: Detailed solutions to instance sets Instance04, -07, -10, -13, -16 and -21.

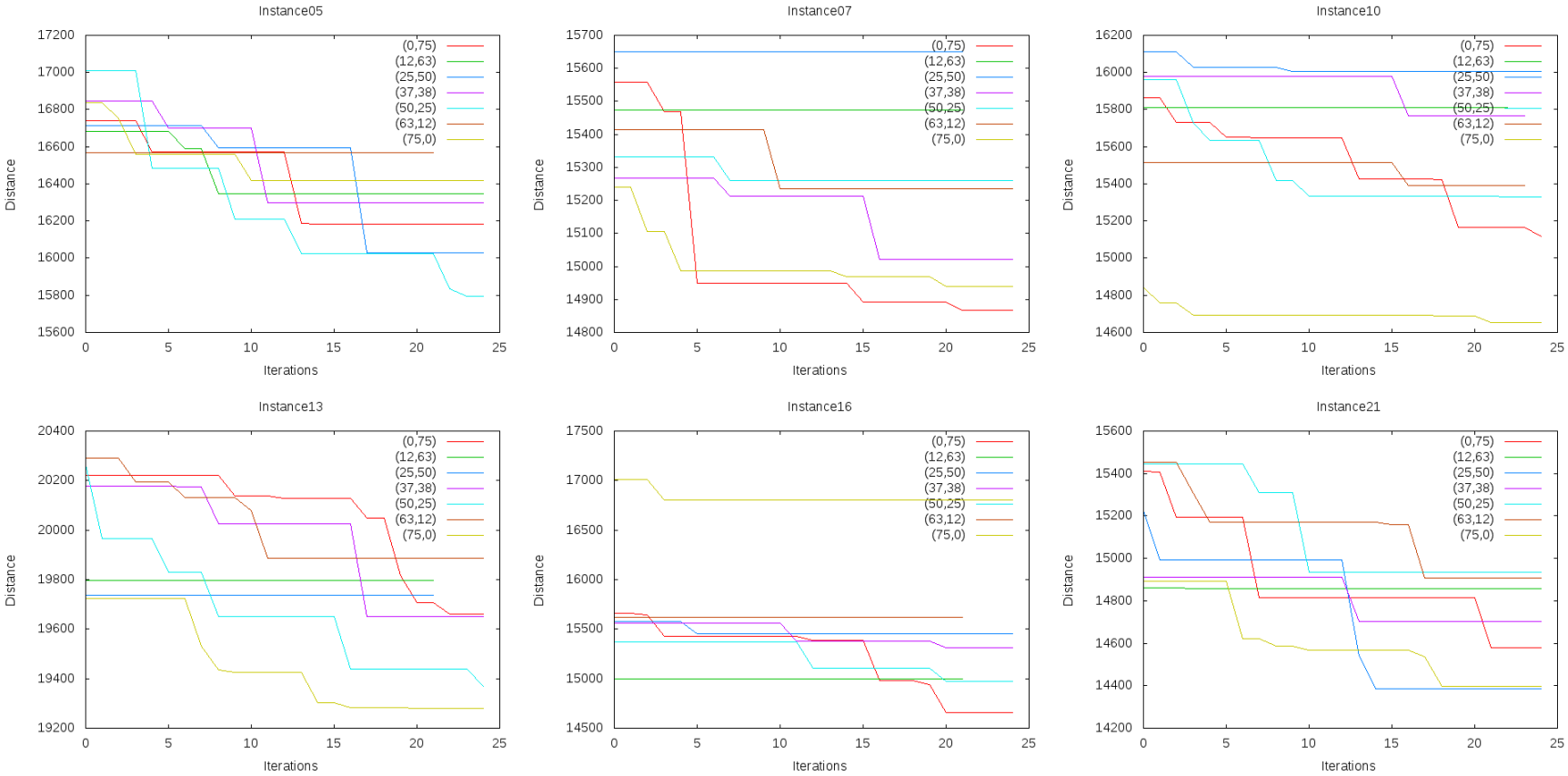


Figure 7.4: Development of the objective of instances with 75 hinterland requests.

7.5 Summary of Findings

The computational results of the exact algorithm and the heuristic algorithm (Chapters 5 and 6) are evaluated by means of three different benchmark sets that strongly vary in size.

To the best of our knowledge, the implementation of the exact approach presented in Chapter 5 (Funke et al. [2016]) is the first implementation of a mathematical model that is able to solve instances of a multi-size container hinterland transportation problem with the option of (de-)coupling containers at customers' locations. The implementation of the mathematical model is able to solve instances regarding two different objectives: minimizing the total travel distance of trucks and minimizing the total operating time of trucks.

On the one hand, the computational results of both algorithms verify the high complexity of computing solutions to instances containing many 20-foot container hinterland requests. This is demonstrated by the fact that the exact approach is able to compute solutions to all but one instance within one hour; the omitted instance contains 20-foot container hinterland requests only (Tables 7.10 and 7.11). Furthermore, sometimes the heuristic approach computes better objective values for instances containing more 40-foot than 20-foot container hinterland requests (Tables 7.10 and 7.11, 7.12 and 7.13). On the other hand, both algorithms need short computation times to compute high-quality solutions to instances containing many 40-foot container hinterland requests. These findings are documented in the columns "ø Gap" and "Σ Time" in Tables 7.11 and A.3. In particular, the tables show that the heuristic approach is often able to compute optimum solutions to those instances. All in all, the results for the different benchmark sets of the heuristic approach deviate from the results of the exact approach by an average of between 2.07% to 7.98%.

The heuristic algorithm is able to compute solutions, whose objective values are close together (Tables 7.12, 7.13 and A.4), to 16 instances out of a total of 20 instances that are inspired by practice. The average computation time of the heuristic approach for an instance inspired by practice is two minutes. The results on the computation time are especially promising as they state that the heuristic approach can be used in practice[7].

[7] As trucking companies are confronted with delays (e.g. traffic, congested terminals) throughout the whole time, a solution methodology might be initiated several times during one workday (refer to Section 1.2.2). Hence, trucking companies typically ask for heuristics computing high-quality solutions within computation times that should not exceed a certain value.

Chapter 8

Conclusion and Future Research

The present thesis investigates a container transportation problem of a trucking company operating in a hinterland region of a seaport. Trucking companies have to transport containerized cargo in a cost-effective manner; the majority of transportation requests are known to trucking companies in advance. Apart from this, trucking companies are responsible for the provision of empty containers to transport cargo; empty containers that share the same attributes (like size) are interchangeable. Hence, the underlying scientific problem comprises two sub-problems: the assignment of empty containers to transportation requests and the construction of routes for trucks. Due to the enormous potential for saving costs, these types of problems have become the subject of many academic investigations. The present thesis analyzes the definition of a widely studied scientific problem definition, in which the definition is extended by two characteristics for which comparatively little literature is available. On the one hand, trucks are permitted to decouple containers at customers' locations for container handling operations (loading and unloading containers). When trucks and containers are allowed to separate, trucks are then free to carry out other tasks while containers are being (un-)loaded. On the other hand, two types of containers are available for transportation, namely 20-foot and 40-foot containers. Although especially 20-foot and 40-foot containers are used in many parts of the world, scientific literature considering the simultaneous transportation of 20-foot and 40-foot containers is rare.

8.1 Conclusion

The key element of the investigation of this thesis is the *multi-size Inland Container Transportation Problem (mICTP)* that is described in detail in Chapter 4. One trucking company owning a homogeneous fleet of trucks has to serve a set of requests for the transportation of containerized cargo and empty containers in a hinterland region of a seaport. The objective is to optimize the costs of the trucking company. In this thesis the costs of the trucking company are measured by the total travel distance of trucks, which should be minimized. The optimization problem includes several decisions that have to be made such as assigning empty containers to transportation requests and constructing routes for trucks that transport the containers. Additionally, containers can be (un-) loaded at customers' locations in a drop-and-pick procedure, i.e. without the presence of a truck. Further decisions have to be taken regarding the separation of the truck and container, and, in cases of separation, regarding the truck that collects the container after it has been (un-)loaded. In this way, the transportation volume and the expense of repositioning empty containers should be decreased. Literature sources commonly discuss modifications on transportation problems only for 40-foot containers (e.g., Braekers et al. [2013], Jula et al. [2005], Nossack and Pesch [2013], Sterzik and Kopfer [2013], Wang and Regan [2002], Zhang et al. [2010]). These problem definitions are representatives of *full truckload (FTL)* problems with trucks being able to transport one 40-foot container at the most. The present thesis enhances the scientific definition of the *Inland Container Transportation Problem (ICT)* (Zhang et al. [2010]), which is widely studied by academic sources (refer to Table 3.1), to the mICTP by considering the simultaneous transportation of containers of different sizes, namely 20-foot and 40-foot containers. The introduction of two different kinds of commodities increases the complexity of the problem considerably: a truck can either transport up to two 20-foot containers or one 40-foot container at a point in time. That is why the problem changes from a FTL problem to a *less-than truckload (LTL)* problem. The mICTP can be classified as a *Vehicle Routing Problem with multiple synchronization constraints (VRPMS, Drexl [2012], Section 3.1.2)*. The considered enlargement of the problem definition is much closer to reality than most previous models that are discussed by literature sources (Schönberger et al. [2013]). Up to now, this modification has rarely been studied in literature sources. Problem definitions for the transportation of 20-foot and 40-foot containers can be found in scientific literature, but mostly these models either prohibit trucks and containers to separate (Chung et al. [2007], Lai et al. [2013], Vidović et al. [2012], Zhang et al. [2015]) leading to routes for trucks containing no more than four hinterland requests, or these models try

to combine special types of hinterland requests (Caballini et al. [2015], Daham et al. [2016]) leading to routes for trucks containing no more than three hinterland requests. The aim of the present thesis is to make a contribution to fill this scientific gap. Models and approaches for the simultaneous transportation of two kinds of commodities are presented for a problem definition allowing trucks and containers to separate during their routes. In this way, it is possible to construct routes for trucks that contain more than four hinterland requests (Schönberger et al. [2013])[1].

Chapter 3 contains comprehensive literature research on container hinterland transportation; the complexity of the mICTP is classified and methodologies are presented to improve the efficiency of drayage including the transportation of containerized cargo and the reduction of unproductive movements caused by empty container repositioning. Mathematical and methodical approaches to the transportation of 40-foot containers are especially at the focus of this literature research (refer to Sections 3.2 and 3.3). This focus is chosen due to the fact that the common problem definition is restricted to the FTL problem for the transportation of a homogeneous set of containers. The aim of the literature research is to enable a better understanding on the complex issue of the thesis. As a result, fast and efficient approaches are combined in order to create a high-level procedure computing solutions to the simultaneous transportation problem of a heterogeneous set of containers in real world applications.

In Chapter 5 the two sub-problems of the mICTP, i.e. assigning empty containers and building routes for trucks, are combined in one single mathematical model simultaneously solving the entire problem definition. Due to the problem's complexity, the simultaneous combination of both sub-problems within one single mathematical model was neglected for many years (Sterzik [2013]). A recent graph representation is developed and combined with successful techniques for modeling hinterland transportation problems (Jula et al. [2005], Nossack and Pesch [2013], Zhang et al. [2010]) and for modeling problems arising in distinct applications (Goel and Meisel [2013]). A *Multicommodity Flow Problem (MFP)* creates flows for containers. For the construction of routes for trucks, we follow the ideas of Jula et al. [2005], Wang and Regan [2002] and transform the *Full Truckload Pickup and Delivery Problem (FTPDPTW)* to the *Asymmetric Multiple Traveling Salesman Problem with Time Window Constraints (amTSPTW)*. Finally, the models for the MFP and the amTSPTW are connected by synchronization constraints. The mathematical model is implemented with the help

[1] Although the model of Schönberger et al. [2013] allow larger routes for trucks. The hinterland requests of the problem definition of Schönberger et al. [2013] are distinct from the hinterland requests contained in the mICTP. Moreover, the implementation of the mathematical model of Schönberger et al. [2013] often fails to even identify a feasible solution.

of a commercial solver. The implementation is capable of computing optimum solutions to small-sized test instances, containing up to eleven hinterland requests and five trucks (Chapter 7). To the best of our knowledge, this is the first implementation of a mathematical model that is able to compute optimum solutions to this complex problem definition containing such a magnitude of different degrees of freedom.

Chapter 6 extends the theoretical considerations that were made so far. As the implementation of the mathematical model takes a considerable amount of time to compute solutions to larger instances, a modern heuristic approach is developed to solve instances arising in real-world applications. Several literature sources (Drexl [2012], Drexl et al. [2011], Meisel and Kopfer [2014]) successfully apply *Large Neighborhood Search (LNS, Ropke and Pisinger [2006])* to VRPMS. Therefore, we combine LNS with mathematical programming techniques. In this manner, we construct a matheuristic approach. Matheuristics are newly developed approaches, in which mathematical programming models are used in heuristic frameworks (see e.g. Archetti and Speranza [2014], Ball [2011] for surveys on matheuristics). The presented heuristic approach combines two phases (Braekers et al. [2013], Caris and Janssens [2009], Imai et al. [2007], Nossack and Pesch [2013]); mathematical models assign empty containers to transportation requests and heuristics build routes for trucks. The potential of the heuristic approach is analyzed two-fold. On the one hand, the solution quality is investigated by comparing solutions to instances taken from literature sources with optimum solutions computed by the implementation of the mathematical model (Tables 7.10 and 7.11); depending on the instances, deviations to optimum solutions vary from approximately two to eight percent. On the other hand, the applicability of the heuristic approach is examined on instances inspired by real-world problems (Chapter 7); these instances contain 75 hinterland requests and 56 trucks and can be solved by the heuristic algorithm within an average computation time of two minutes.

8.2 Future Research

The present thesis introduces a very recent definition of a drayage problem to the reader. Up to now, both the simultaneous transportation problem of 20-foot and 40-foot containers as well as the possibility to separate trucks and containers at customers' locations has been rarely investigated in literature sources. As a result, this topic remains interesting for future research. Furthermore, some of the results of this thesis might give rise to further scientific issues. In the following some interesting aspects are mentioned that might be suitable for

deeper analysis. Possible extensions accrue in one of two ways: algorithmic and problem-specific.

On the algorithmic side, the mathematical approach can be extended by linear and mathematical programming techniques (like decomposition) in order to solve instances of larger sizes. By comparison, the heuristic approach can be made adaptive by including different mechanisms considering the performances of former iterations of the selection of destroy and repair operators, the use of noise and tabu techniques, as well as the choice of parts of the solution for re-optimization. The computational study indicates that instances including the same number of hinterland requests are harder to solve for both approaches when the set of hinterland requests contains more hinterland requests for 20-foot containers than hinterland requests for 40-foot containers. As the solution space enlarges when the number of 20-foot container hinterland requests grows, the computation time of the implementation of the mathematical model increases. Another reason for the complexity of solving these instances is the insufficient progress in the research area on problems containing double loads to date. The solution quality of the heuristic approach may improve, if for example recent local search operators are developed that are specifically designed for the handling of 20-foot container hinterland requests. Finally, a further approach has to be developed for the computation of lower bounds on instances that, due to their size, could not be solved by the implementation of the mathematical model. Up until now, there has only been one way to evaluate solutions to large size instances that are computed by the heuristic approach; solutions to these instances have to be compared with other solutions to similar instances that are also obtained by the heuristic approach (refer e.g. to Section 7.4.4, in which we compare solutions to instances that differ in the number of 40-foot and 20-foot container hinterland requests).

As can be seen from Section 3.5 various interesting challenges arise in the research area on container hinterland transportation. Consequently, a lot of different restrictions and extensions can be added to the mICTP in order to make the problem definition become even more complex and realistic. As shown in Table 3.1, the mICTP integrates all but one of the headings that are analyzed in more depth in the literature overview in Chapter 3. The inclusion of a heterogeneous fleet of trucks is left out in the definition of the mICTP, i.e. there is no distinction made between different costs of trucks (company-owned and mandated trucks), or between the trucks' capacities (combined, 20-foot and 40-foot trucks). Moreover, the set of containers can be further distinguished into dry and special containers; dependencies and exclusions between truck types and container types can be investigated (refer to Section 1.1). The costs for detention of containers (Caballini et al. [2015]) might be added to the problem

definition, as well as vessel departure times and the additional consideration of soft time windows, instead of hard time windows at the different locations. Within our extensive literature review, we found only two literature sources (Reinhardt et al. [2012], Zhang et al. [2009]) considering a limitation/approach to balancing the number of empty containers that can be stored at container depots. Thus, a further challenging issue is to analyze problems, in which only a limited number of empty containers can be stored at depots. In this context it is interesting to allow the substitution of containers, i.e. serving hinterland requests for 20-foot containers by either 20-foot or 40-foot containers, or allowing some hinterland requests for 40-foot containers to also be served by two 20-foot containers, instead of one 40-foot container. This issue is investigated by Engels and Schrader [2015], who optimize the transportation costs of a railway company that is responsible for the transportation of goods between different customers' locations. In more detail, Engels and Schrader [2015] establish the enormous complexity that is achieved when integrating the possibility of substitution of freight cars into a problem definition asking for a minimum cost assignment of freight cars to customers' transportation requests. In addition, the two special characteristics of the mICTP can be further extended. Regarding the possibility to decouple containers at customers' locations, it might be useful to distinguish between two types of customers: those customers, who have the opportunity to lift containers from trucks and those customers, who do not have this opportunity. Up until now, the mICTP has not considered the order, in which 20-foot containers are coupled by trucks. The problem definition might become more realistic when routes for 20-foot containers are constructed in a *Last In Last Out (LIFO) technique*, i.e. the last container coupled by a truck is restricted to be the first container to be decoupled from the truck. To overcome this issue, Zhang et al. [2015] introduce the *full-twin assumption*, in which trucks carrying two 20-foot containers have to complete both requests before a new request is initiated. The full-twin assumption, however, restricts two hinterland requests to be served without any interruption. This restriction is much stricter than allowing requests to be served within a truck's route at any point of time that observes the request's time windows, as the case is in this research project.

Appendix A

Further Computational Results

This chapter provides the remaining results that are not listed in Chapter 7. The structure of the appendices is as follows:

- Tables A.1 and A.2 contain detailed solutions of the exact approach to the randomly generated instance set containing up to six hinterland requests. These results extend Section 7.3.
- Table A.3 contains results that are not mentioned in detail in Section 7.4.3. The results refer to the adjusted benchmark set of the instances of Sterzik and Kopfer [2013].
- Further results of the instance set, which is an adjustment of the instance set of Zhang et al. [2010], are summarized in Table A.4 and Figures A.1 and A.2. These results are excluded in Section 7.4.4.

Set, 5min	Exact, Objective 5.27				Exact, Objective 5.28						
	Opt	Inf	Km	Time	Opt	Feas	Inf	Serv	Km	Gap	Time
(0, 6)	19	2	28713.5	6.38	16	3	2	38414.1	30053.1	79	16547.00
(0, 6), 1_truck	5	2	28156.2	5.81	5	0	2	33694.4	28156.2	0	809.37
(0, 6), 2_truck	7	0	28912.6	0.27	7	0	0	42585.9	30124.6	0	4187.22
(0, 6), 3_truck	7	0	28912.6	0.30	4	3	0	37613.6	31336.6	213	11550.40
(0, 6), no_tw	3	0	26438.0	0.23	3	0	0	28838.0	26438.0	0	876.64
(0, 6), rand_tw	9	0	26488.3	0.42	6	3	0	28888.3	26488.3	166	15298.70
(0, 6), real_tw	7	2	32549.7	5.73	7	0	2	54765.6	36185.7	0	371.75
(2, 2)	19	2	26705.2	5.30	19	0	2	37331.9	29829.4	0	4843.08
(2, 2), 1_truck	5	2	26927.2	2.34	5	0	2	37150.8	30316.2	0	267.84
(2, 2), 2_truck	7	0	26625.9	1.52	7	0	0	39982.7	29857.1	0	873.00
(2, 2), 3_truck	7	0	26625.9	1.44	7	0	0	34810.6	29453.9	0	3702.24
(2, 2), no_tw	3	0	24433.0	2.48	3	0	0	30384.0	24433.0	0	913.24
(2, 2), rand_tw	9	0	24433.0	0.77	9	0	0	30384.0	24433.0	0	3737.82
(2, 2), real_tw	7	2	30600.3	2.05	7	0	2	49242.7	39080.3	0	192.02
(1, 2)	21	0	8484.0	2.11	21	0	0	13440.2	9292.0	0	22.95
(1, 2), 1_truck	7	0	8484.0	0.67	7	0	0	15048.6	8484.0	0	5.50
(1, 2), 2_truck	7	0	8484.0	0.78	7	0	0	12635.7	9696.0	0	6.58
(1, 2), 3_truck	7	0	8484.0	0.66	7	0	0	12636.3	9696.0	0	10.87
(1, 2), no_tw	3	0	8484.0	1.45	3	0	0	11424.0	8484.0	0	10.25
(1, 2), rand_tw	9	0	8484.0	0.40	9	0	0	11423.8	8484.0	0	7.50
(1, 2), real_tw	9	0	8484.0	0.26	9	0	0	16128.7	10369.3	0	5.20
(1, 3)	21	0	24104.0	2.03	21	0	0	30982.0	24912.0	0	80.24
(1, 3), 1_truck	7	0	24104.0	0.69	7	0	0	35699.3	24912.0	0	12.96
(1, 3), 2_truck	7	0	24104.0	0.61	7	0	0	28623.6	24912.0	0	32.23
(1, 3), 3_truck	7	0	24104.0	0.73	7	0	0	28623.0	24912.0	0	35.05
(1, 3), no_tw	3	0	24104.0	1.32	3	0	0	27644.0	24104.0	0	32.00
(1, 3), rand_tw	9	0	24104.0	0.43	9	0	0	27643.7	24104.0	0	42.03
(1, 3), real_tw	9	0	24104.0	0.28	9	0	0	35432.9	25989.3	0	6.21

Table A.1: Detailed solutions to instances with up to six hinterland requests and a (de-)couple duration of five minutes, exact approach (Funke and Kopfer [2016]).

Set, 20min	Exact, Objective 5.27				Exact, Objective 5.28						
	Opt	Inf	Km	Time	Opt	Feas	Inf	Serv	Km	Gap	Time
(0, 6)	18	3	28422.8	4.22	15	3	3	46437.9	30372.8	68	14537.70
(0, 6), 1_truck	4	3	26551.5	3.53	4	0	3	36751.8	26551.5	0	639.69
(0, 6), 2_truck	7	0	28957.4	0.30	7	0	0	52286.3	30169.4	0	2340.86
(0, 6), 3_truck	7	0	28957.4	0.39	4	3	0	46124.4	32759.7	174	11557.10
(0, 6), no_tw	3	0	26438.0	0.19	3	0	0	36038.0	26438.0	0	1318.43
(0, 6), rand_tw	9	0	26589.3	0.54	6	3	0	36989.6	26589.3	135	13022.90
(0, 6), real_tw	6	3	32165.3	3.49	6	0	3	65810.3	38015.3	0	196.34
(2, 2)	18	3	26138.6	5.79	18	0	3	42308.6	28338.1	0	4762.19
(2, 2), 1_truck	4	3	24433.0	2.91	4	0	3	35784.0	24433.0	0	207.49
(2, 2), 2_truck	7	0	26625.9	1.62	7	0	0	47106.6	29453.9	0	915.72
(2, 2), 3_truck	7	0	26625.9	1.26	7	0	0	41238.9	29453.9	0	3638.98
(2, 2), no_tw	3	0	24433.0	2.47	3	0	0	35784.0	24433.0	0	914.25
(2, 2), rand_tw	9	0	24433.0	0.76	9	0	0	35784.2	24433.0	0	3744.17
(2, 2), real_tw	6	3	29549.7	2.56	6	0	3	55357.3	36148.3	0	103.77
(1, 2)	21	0	8484.0	2.36	21	0	0	16954.3	9696.0	0	18.89
(1, 2), 1_truck	7	0	8484.0	0.73	7	0	0	18391.3	9696.0	0	5.33
(1, 2), 2_truck	7	0	8484.0	0.75	7	0	0	16235.9	9696.0	0	6.81
(1, 2), 3_truck	7	0	8484.0	0.88	7	0	0	16235.7	9696.0	0	6.75
(1, 2), no_tw	3	0	8484.0	1.57	3	0	0	15023.7	8484.0	0	7.18
(1, 2), rand_tw	9	0	8484.0	0.42	9	0	0	15023.9	8484.0	0	7.74
(1, 2), real_tw	9	0	8484.0	0.37	9	0	0	19528.2	11312.0	0	3.97
(1, 3)	19	2	24104.0	2.34	19	0	2	36432.3	25591.3	0	77.71
(1, 3), 1_truck	5	2	24104.0	0.92	5	0	2	36479.2	24104.0	0	13.95
(1, 3), 2_truck	7	0	24104.0	0.54	7	0	0	38294.4	27332.9	0	24.25
(1, 3), 3_truck	7	0	24104.0	0.88	7	0	0	34536.7	24912.0	0	39.51
(1, 3), no_tw	3	0	24104.0	1.34	3	0	0	33044.0	24104.0	0	33.46
(1, 3), rand_tw	9	0	24104.0	0.41	9	0	0	33043.8	24104.0	0	39.01
(1, 3), real_tw	7	2	24104.0	0.59	7	0	2	42241.1	28140.9	0	5.24

Table A.2: Detailed solutions to instances with up to six hinterland requests and a (de-)couple duration of twenty minutes, exact approach (Funke and Kopfer [2016]).

Set	Exact						LNS		
	Opt	Feas	Inf	ø Km	ø Gap	Σ Time	ø Km	ø Gap	Σ Time
DS1, all	0	6	0	485.000	3.88	2457.70	512.000	5.57	172.35
DS1, (0, 11)	0	1	0	408	5.00	2451.29	503	23.28	79.33
DS1, (3, 8)	0	1	0	458	4.27	5.31	525	14.63	4.46
DS1, (5, 6)	0	1	0	473	4.23	0.50	473	0.00	58.41
DS1, (6, 5)	0	1	0	543	4.86	0.40	543	0.00	19.23
DS1, (8, 3)	0	1	0	525	0.94	0.18	525	0.00	9.71
DS1, (11, 0)	0	1	0	503	3.98	0.02	503	0.00	1.21
DS3, all	0	6	0	719.500	8.41	8682.15	748.000	3.96	145.81
DS3, (0, 11)	0	1	0	638	23.11	3599.21	659	3.29	28.48
DS3, (3, 8)	0	1	0	767	11.83	3599.80	773	0.78	24.27
DS3, (5, 6)	0	1	0	741	5.00	1480.23	783	5.67	3.93
DS3, (6, 5)	0	1	0	741	4.99	2.67	745	0.54	81.50
DS3, (8, 3)	0	1	0	771	2.59	0.21	771	0.00	6.50
DS3, (11, 0)	0	1	0	659	2.92	0.03	757	14.87	1.13
DS4, all	1	5	0	620.500	9.86	7327.59	664.833	7.14	163.57
DS4, (0, 11)	0	1	0	614	36.50	3598.83	625	1.79	131.35
DS4, (3, 8)	0	1	0	582	9.51	3599.81	625	7.39	3.35
DS4, (5, 6)	0	1	0	616	4.94	25.39	702	13.96	7.01
DS4, (6, 5)	0	1	0	659	5.00	103.29	777	17.91	3.58
DS4, (8, 3)	0	1	0	627	3.19	0.23	630	0.48	16.92
DS4, (11, 0)	1	0	0	625	0.00	0.04	630	0.80	1.37
DS5, all	1	5	0	616.833	5.44	7205.89	645.500	4.65	186.25
DS5, (0, 11)	0	1	0	573	10.59	3598.82	586	2.27	7.37
DS5, (3, 8)	0	1	0	651	8.71	3599.95	731	12.29	37.01
DS5, (5, 6)	0	1	0	618	4.26	0.81	631	2.10	66.66
DS5, (6, 5)	0	1	0	631	4.98	6.20	631	0.00	65.55
DS5, (8, 3)	0	1	0	655	4.12	0.10	721	10.08	8.07
DS5, (11, 0)	1	0	0	573	0.00	0.01	573	0.00	1.59
DS6, all	1	5	0	606.500	8.95	7236.40	653.500	7.75	182.56
DS6, (0, 11)	0	1	0	587	23.51	3598.62	710	20.95	3.99
DS6, (3, 8)	0	1	0	632	15.51	3600.00	647	2.37	70.63
DS6, (5, 6)	0	1	0	587	5.00	34.94	594	1.19	34.87
DS6, (6, 5)	0	1	0	593	4.72	2.34	602	1.52	65.61
DS6, (8, 3)	0	1	0	647	4.94	0.47	647	0.00	6.52
DS6, (11, 0)	1	0	0	593	0.00	0.03	721	21.59	0.94
DS9, all	0	6	0	574.000	5.72	7216.89	614.667	7.08	342.27
DS9, (0, 11)	0	1	0	530	12.33	3598.00	541	2.08	47.16
DS9, (3, 8)	0	1	0	594	5.22	3599.80	607	2.19	92.45
DS9, (5, 6)	0	1	0	605	4.98	4.15	671	10.91	189.88
DS9, (6, 5)	0	1	0	548	5.00	14.69	610	11.31	6.82
DS9, (8, 3)	0	1	0	575	3.44	0.22	606	5.39	4.65
DS9, (11, 0)	0	1	0	592	3.38	0.03	653	10.30	1.31
DS10, all	1	5	0	742.000	7.97	7862.42	788.500	6.27	97.95
DS10, (0, 11)	0	1	0	680	26.11	3598.85	773	13.68	23.92
DS10, (3, 8)	0	1	0	790	11.27	3599.89	790	0.00	25.13
DS10, (5, 6)	0	1	0	756	5.00	655.89	773	2.25	26.98
DS10, (6, 5)	0	1	0	756	5.00	7.56	832	10.05	17.76
DS10, (8, 3)	1	0	0	790	0.00	0.21	807	2.15	2.74
DS10, (11, 0)	0	1	0	680	0.44	0.02	756	11.18	1.43

Table A.3: Detailed solutions to instance sets DS1, DS3, . . . DS6, DS9 and DS10.

Set	ø $\lvert\mathcal{T}\rvert$	ø Km	Σ Time	Set	ø $\lvert\mathcal{T}\rvert$	ø Km	Σ Time
Instance05, all	46.14	16233.7	795.646	Instance15, all	43.57	16298.6	599.465
Instance05, (0, 75)	46	16185	89.205	Instance15, (0, 75)	42	16069	155.929
Instance05, (12, 63)	46	16346	114.381	Instance15, (12, 63)	43	15531	113.043
Instance05, (25, 50)	47	16028	85.138	Instance15, (25, 50)	41	16294	35.248
Instance05, (37, 38)	46	16298	112.018	Instance15, (37, 38)	42	15956	103.026
Instance05, (50, 25)	46	15794	112.797	Instance15, (50, 25)	42	16250	67.518
Instance05, (63, 12)	46	16568	89.372	Instance15, (63, 12)	42	16343	80.122
Instance05, (75, 0)	46	16417	192.735	Instance15, (75, 0)	53	17647	44.580
Instance06, all	47.43	14150.9	791.200	Instance17, all	43.57	16370	509.057
Instance06, (0, 75)	48	14472	151.761	Instance17, (0, 75)	43	16303	137.857
Instance06, (12, 63)	48	13932	144.015	Instance17, (12, 63)	41	16211	94.279
Instance06, (25, 50)	47	14373	67.256	Instance17, (25, 50)	43	16356	31.228
Instance06, (37, 38)	50	13961	57.795	Instance17, (37, 38)	45	16139	102.417
Instance06, (50, 25)	47	14304	75.921	Instance17, (50, 25)	42	15893	19.104
Instance06, (63, 12)	47	14120	102.727	Instance17, (63, 12)	41	16416	33.882
Instance06, (75, 0)	45	13894	191.725	Instance17, (75, 0)	50	17272	90.290
Instance08, all	43.14	13688.6	703.669	Instance18, all	44.00	14866.6	831.756
Instance08, (0, 75)	42	13482	94.645	Instance18, (0, 75)	45	14960	97.576
Instance08, (12, 63)	45	13376	1472.000	Instance18, (12, 63)	43	14511	90.272
Instance08, (25, 50)	43	13760	94.271	Instance18, (25, 50)	44	14706	82.007
Instance08, (37, 38)	43	13400	82.454	Instance18, (37, 38)	43	14425	75.476
Instance08, (50, 25)	43	14123	25.574	Instance18, (50, 25)	45	15196	134.175
Instance08, (63, 12)	44	14032	124.315	Instance18, (63, 12)	40	14814	280.901
Instance08, (75, 0)	42	13647	135.408	Instance18, (75, 0)	48	15454	71.350
Instance09, all	45.71	18509.6	623.056	Instance19, all	39.29	17554.9	532.980
Instance09, (0, 75)	47	18164	108.997	Instance19, (0, 75)	37	17379	111.530
Instance09, (12, 63)	45	18861	65.037	Instance19, (12, 63)	38	17111	40.515
Instance09, (25, 50)	45	18743	50.199	Instance19, (25, 50)	37	17182	102.599
Instance09, (37, 38)	47	18861	28.865	Instance19, (37, 38)	35	17585	81.155
Instance09, (50, 25)	46	18365	79.758	Instance19, (50, 25)	37	18099	71.709
Instance09, (63, 12)	46	18910	167.674	Instance19, (63, 12)	36	16600	84.193
Instance09, (75, 0)	44	17663	122.526	Instance19, (75, 0)	55	18928	41.279
Instance11, all	48.29	16725.3	682.326	Instance20, all	29.57	14087.6	1673.370
Instance11, (0, 75)	49	16896	41.153	Instance20, (0, 75)	30	14456	238.735
Instance11, (12, 63)	48	16552	61.237	Instance20, (12, 63)	29	14139	295.695
Instance11, (25, 50)	49	16893	59.253	Instance20, (25, 50)	30	13872	198.944
Instance11, (37, 38)	50	17135	135.604	Instance20, (37, 38)	30	13733	165.533
Instance11, (50, 25)	46	16586	119.107	Instance20, (50, 25)	31	14838	169.697
Instance11, (63, 12)	48	16984	106.740	Instance20, (63, 12)	30	14787	120.800
Instance11, (75, 0)	48	16031	159.232	Instance20, (75, 0)	27	12788	483.962
Instance12, all	41.43	13636.3	705.419	Instance22, all	36.14	17426.1	911.985
Instance12, (0, 75)	42	14500	79.716	Instance22, (0, 75)	34	16751	338.959
Instance12, (12, 63)	43	13881	66.792	Instance22, (12, 63)	35	17725	57.071
Instance12, (25, 50)	42	13687	72.359	Instance22, (25, 50)	34	17481	119.598
Instance12, (37, 38)	41	13118	69.672	Instance22, (37, 38)	35	17650	44.828
Instance12, (50, 25)	42	14019	83.063	Instance22, (50, 25)	36	17125	201.490
Instance12, (63, 12)	41	13837	103.133	Instance22, (63, 12)	35	17565	75.545
Instance12, (75, 0)	39	12412	230.686	Instance22, (75, 0)	44	17686	74.494
Instance14, all	40.14	15595.6	1179.810	Instance23, all	34.14	15108.3	2173.470
Instance14, (0, 75)	37	15893	264.451	Instance23, (0, 75)	35	15910	154.375
Instance14, (12, 63)	38	15726	238.256	Instance23, (12, 63)	35	14478	190.211
Instance14, (25, 50)	36	15195	107.121	Instance23, (25, 50)	34	15844	136.639
Instance14, (37, 38)	39	15264	194.811	Instance23, (37, 38)	34	15898	115.995
Instance14, (50, 25)	38	15866	96.653	Instance23, (50, 25)	35	15395	296.564
Instance14, (63, 12)	39	14691	190.475	Instance23, (63, 12)	35	14824	470.185
Instance14, (75, 0)	54	16534	88.041	Instance23, (75, 0)	31	13409	809.500

Table A.4: Detailed solutions to further instances with 75 hinterland requests.

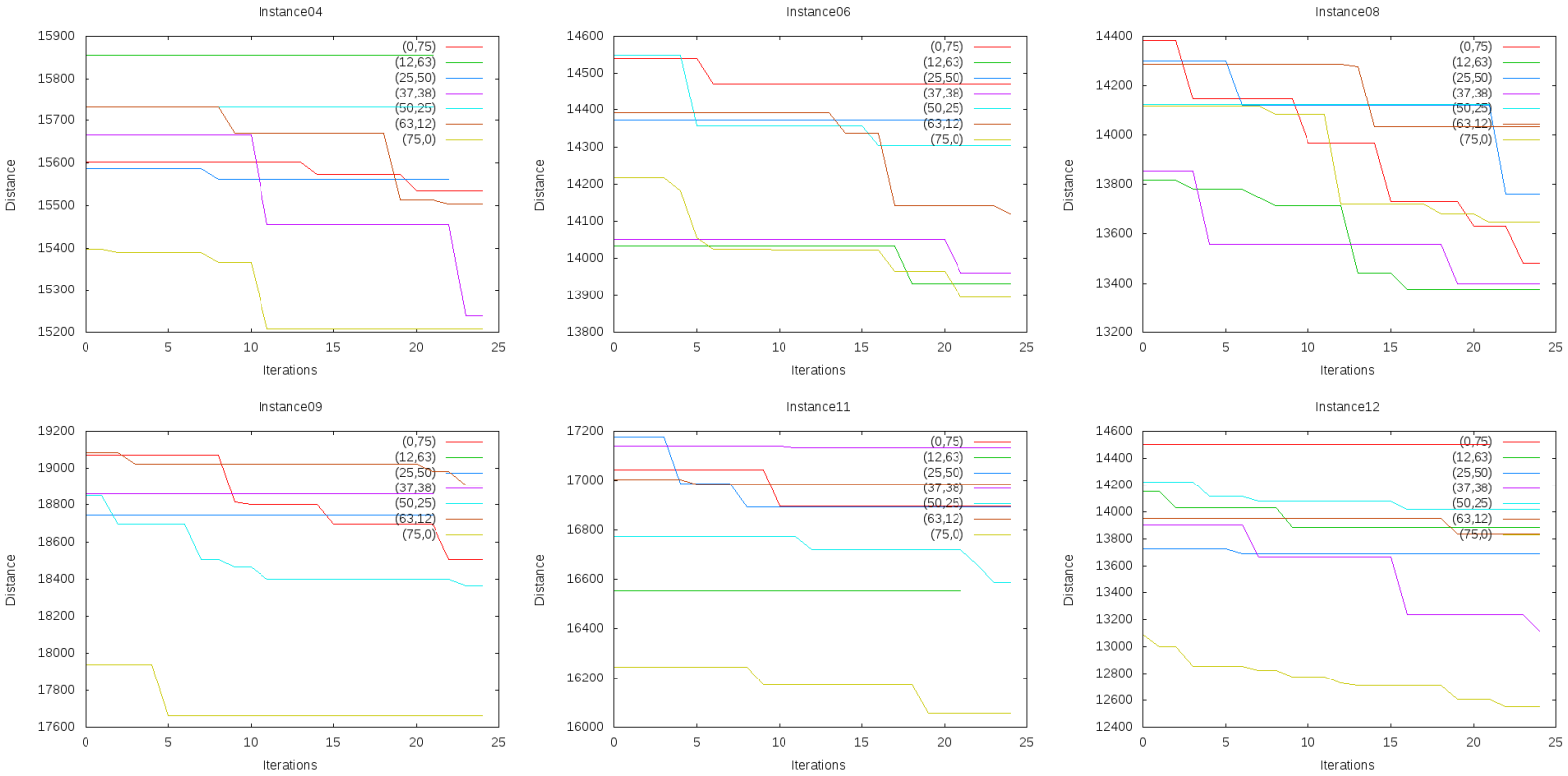


Figure A.1: Development of the objective of instances with 75 hinterland requests.

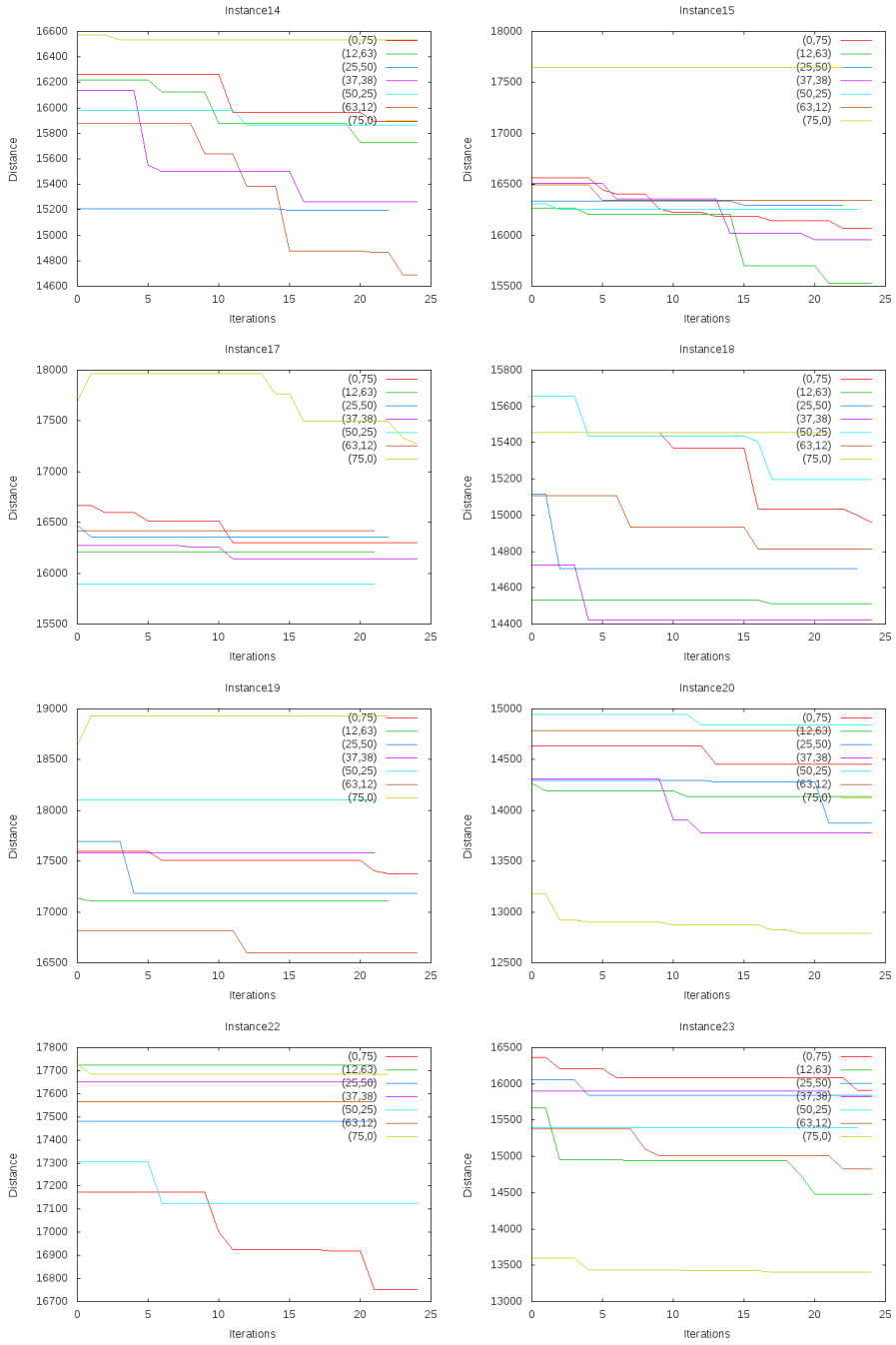


Figure A.2: Development of the objective of instances with 75 hinterland requests.

Bibliography

K. Alisch, E. Winter, and U. Arentzen. *Gabler Wirtschaftslexikon*. Springer-Verlag, 2013.

D. Applegate, R. E. Bixby, V. Chvátal, and W. Cook. *Finding cuts in the TSP (A preliminary report)*, volume 95-05. Citeseer, 1995.

D. Applegate, R. E. Bixby, V. Chvátal, and W. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton university press, 2011.

C. Archetti and M. G. Speranza. A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, 2(4):223–246, 2014.

L. Asbach, U. Dorndorf, and E. Pesch. Analysis, modeling and solution of the concrete delivery problem. *European journal of operational research*, 193(3): 820–835, 2009.

M. O. Ball. Heuristics based on mathematical programming. *Surveys in Operations Research and Management Science*, 16(1):21 – 38, 2011.

R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA journal on computing*, 6(2):126–140, 1994.

T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

R. Bellman. On a routing problem. *Quarterly of applied mathematics*, pages 87–90, 1958.

R. Bent and P. Van Hentenryck. A two-phase hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Comput Oper Res*, 33 (4):875–893, 2006.

G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *TOP*, 15(1):1–31, 2007. ISSN 1134-5764.

C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

L. Bodin, A. Mingozzi, R. Baldacci, and M. Ball. The rollon–rolloff vehicle routing problem. *Transportation Science*, 34(3):271–288, 2000.

M. Boile, N. Mittal, M. Golias, and S. Theofanis. Empty marine container management: addressing a global problem locally. In *Transportation Research Board 85th Annual Meeting*, volume 06-2147, 2006.

K. Braekers, A. Caris, and G. Janssens. A deterministic annealing algorithm for simultaneous routing of loaded and empty containers. In G. Lencse and Muka L., editors, *Proceedings of the ISC'2010 (Industrial Simulation Conference)*, pages 172–178, Budapest, Hungary, 2010.

K. Braekers, G. K. Janssens, and A. Caris. Challenges in managing empty container movements at multiple planning levels. *Transport Reviews*, 31(6):681–708, 2011.

K. Braekers, A. Caris, and G. K. Janssens. Integrated planning of loaded and empty container movements. *OR Spectrum*, 35(2):457–478, 2013.

A. E. Branch. *Export Practice and Management*. (London: Thomson Learning), 2006.

O. Bräysy, W. Dullaert, G. Hasle, D. Mester, and M. Gendreau. An effective multirestart deterministic annealing metaheuristic for the fleet size and mix vehicle-routing problem with time windows. *Transportation Science*, 42(3): 371–386, 2008.

C. Caballini, I. Rebecchi, and S. Sacone. Combining multiple trips in a port environment for empty movements minimization. *Transportation Research Procedia*, 10:694–703, 2015.

A. Caris and G. K. Janssens. A local search heuristic for the pre-and end-haulage of intermodal container terminals. *Computers & Operations Research*, 36(10):2763–2772, 2009.

A. Caris and G. K. Janssens. A deterministic annealing algorithm for the pre- and end-haulage of intermodal container terminals. *International Journal of Computer Aided Engineering and Technology*, 2(4):340–355, 2010.

V. Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

R. K. Cheung, N. Shi, W. B. Powell, and H. P. Simao. An attribute–decision model for cross-border drayage problem. *Transportation Research Part E: Logistics and Transportation Review*, 44(2):217–234, 2008.

K. H. Chung, C.S. Ko, J.Y. Shin, H. Hwang, and K.H. Kim. Development of mathematical models for the container road transportation in korean trucking industries. *Computers & Industrial Engineering*, 53(2):252 – 262, 2007.

G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

L. Coslovich, R. Pesenti, and W. Ukovich. Minimizing fleet operating costs for a container transportation company. *European Journal of Operational Research*, 171(3):776–786, 2006.

T. G. Crainic and G. Laporte. Planning models for freight transportation. *European journal of operational research*, 97(3):409–438, 1997.

T. G. Crainic, M. Gendreau, and P. Dejax. Dynamic and stochastic models for the allocation of empty containers. *Operations research*, 41(1):102–126, 1993.

R. H. Currie and S. Salhi. Exact and heuristic methods for a full-load, multiterminal, vehicle scheduling problem with backhauling and time windows. *Journal of the Operational Research Society*, 54(4):390–400, 2003.

H. A. Daham, X. Yang, and M. K. Warnes. An efficient mixed integer programming model for pairing containers in inland transportation based on the assignment of orders. *Journal of the Operational Research Society*, pages 1–17, 2016.

G. Dantzig and D. R. Fulkerson. On the max flow min cut theorem of networks. In H. W. Kuhn and A. W. Tucker, editors, *Linear inequalities and related systems*, pages 215–221, Princeton, 1956. University Press.

R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1):61–95, 1991.

M. Desroches, J. K. Lenstra, , M. W. P. Savelsbergh, and F. Soumis. Vehicle routing with time windows: optimization and approximation. *Vehicle routing: Methods and studies*, pages 65–84, 1988.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. ISSN 0029-599X. doi: 10.1007/BF01386390.

R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, pages 161–166, 1950.

K. F. Doerner and V. Schmid. Survey: matheuristics for rich vehicle routing problems. In *International Workshop on Hybrid Metaheuristics*, pages 206–221. Springer, 2010.

M. Drexl. Synchronization in vehicle routing—a survey of vrps with multiple synchronization constraints. Technical Report 1103, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, 2011.

M. Drexl. Synchronization in vehicle routing-a survey of vrps with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012.

M. Drexl, J. Rieck, T. Sigl, and B. Berning. Simultaneous vehicle and crew routing and scheduling for partial and full load long-distance road transport. Technical Report 1112, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, 2011.

G. Dueck and T. Scheuer. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of computational physics*, 90(1):161–175, 1990.

U. Duken and Gesamtverband der Deutschen Versicherungswirtschaft. Containerhandbuch:[fachinformation der deutschen transportversicherer]. `http://www.containerhandbuch.de/`, 2002-2016. Accessed on 1st March 2016.

M. Eiglsperger. *Automatic layout of UML class diagrams: a topology-shape-metrics approach*. PhD thesis, Universität Tübingen, 2003.

B. Engels and R. Schrader. Freight car dispatching with generalized flows. *Networks*, 66(1):33–39, 2015.

A. L. Erera and K. R. Smilowitz. Intermodal drayage routing and scheduling. *Intelligent Freight Transportation, automation and control engineering series*, pages 171–188, 2008.

S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5:691–703, 1976.

R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

L. R. Ford. Network flow theory. Technical report, DTIC Document, 1956.

L. R. Ford and D. R. Fulkerson. Solving the transportation problem. *Management Science*, 3(1):24–32, 1956.

L. R. Ford and D. R. Fulkerson. Flows in Networks. *Princeton University Press, Princeton*, 1962.

P. Francis, G. Zhang, and K. Smilowitz. Improved modeling and solution methods for the multi-resource routing problem. *European Journal of Operational Research*, 180(3):1045–1059, 2007.

M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

D. R. Fulkerson. Note on dilworth's decomposition theorem for partially ordered sets. In *Proc. Amer. Math. Soc*, volume 7, pages 701–702, 1956.

J. Funke and H. Kopfer. A model for an inland multi-size container transportation problem. Chair of Logistics Working Paper No. 7, University of Bremen, 2014.

J. Funke and H. Kopfer. A neighborhood search for a multi-size container transportation problem. *15th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2015, IFAC-PapersOnLine*, 48(3):2041–2046, 2015.

J. Funke and H. Kopfer. A model for a multi-size inland container transportation problem. *Transportation Research Part E: Logistics and Transportation Review*, 89:70–85, 2016.

J. Funke, S. Hougardy, and J. Schneider. An exact algorithm for wirelength optimal placements in VLSI design. *Integration, the VLSI Journal*, 52:355–366, 2016.

B. Gavish and K. Srikanth. An optimal solution method for large-scale multiple traveling salesmen problems. *Operations Research*, 34(5):698–717, 1986.

M. Gendreau. An introduction to tabu search. In *Handbook of metaheuristics*, pages 37–54. Springer, 2003.

B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, 22(2):340–349, 1974.

F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

A. Goel and F. Meisel. Workforce routing and scheduling for electricity network maintenance with downtime minimization. *European Journal of Operational Research*, 231(1):210–228, 2013.

M. Grötschel. Schnelle Rundreisen: Das Travelling-Salesman-Problem. In *Diskrete Mathematik erleben*, pages 95–129. Springer, 2015.

H. O. Günther and K. H. Kim. Container terminals and terminal operations. *OR Spectrum*, 28(4):437–445, 2006. ISSN 0171-6468.

K. Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

W. C. Hildebrand. *Management von Transportnetzwerken im containerisierten Seehafenhinterlandverkehr: [ein Gestaltungsmodell zur Effizienzsteigerung von Transportprozessen in der Verkehrslogistik]*. PhD thesis, Technische Universität Berlin, Berlin, 2008.

F. L. Hitchcock. The distribution of a product from several sources to numerous localities. *Journal of mathematics and physics*, 20(1):224–230, 1941.

K. L. Hoffman, M. Padberg, and G. Rinaldi. Traveling salesman problem. In *Encyclopedia of Operations Research and Management Science*, pages 1573–1578. Springer, 2013.

J. Homberger and H. Gehring. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162(1):220–238, 2005.

Y. Ileri, M. Bazaraa, T. Gifford, G. Nemhauser, J. Sokol, and E. Wikum. An optimization approach for planning daily drayage operations. *Central European Journal of Operations Research*, 14(2):141–156, 2006.

A. Imai, E. Nishimura, and J. Current. A lagrangian relaxation-based heuristic for the vehicle routing with full container load. *European Journal of Operational Research*, 176(1):87–105, 2007.

K. Jacobs and D. Jungnickel. *Einführung in die Kombinatorik*. Walter de Gruyter, 2004.

J. J. Jaw, A. R. Odoni, H. N. Psaraftis, and N. H. M. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, 20(3):243–257, 1986.

D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovitch. Experimental analysis of heuristics for the atsp. In *The traveling salesman problem and its variations*, pages 445–487. Springer, 2007.

H. Jula, M. Dessouky, P. Ioannou, and A. Chassiakos. Container movement by trucks in metropolitan networks: modeling and optimization. *Transportation Research Part E: Logistics and Transportation Review*, 41(3):235–259, 2005.

H. Jula, A. Chassiakos, and P. Ioannou. Port dynamic empty container reuse. *Transportation Research Part E: Logistics and Transportation Review*, 42(1):43–60, 2006.

R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller, J .W. Thatcher, and J .D. Bohlinger, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US, 1972.

V. King, S. Rao, and R. Tarjan. A faster deterministic maximum flow algorithm. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 157–164. Society for Industrial and Applied Mathematics, 1992.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

D. König. Graphs and matrices. *Matematikaiés Fizikai Lapok*, 38:116–119, 1931.

D. König. *Theorie der endlichen und unendlichen Graphen*. Chelsea Publishing Co., Leipzig, 1936.

R. Konings. Foldable containers to reduce the costs of empty transport? a cost–benefit analysis from a chain and multi-actor perspective. *Maritime Economics & Logistics*, 7(3):223–249, 2005.

R. Konings and R. Thijs. Foldable containers: a new perspective on reducing container-repositioning costs. *European journal of transport and infrastructure research EJTIR, 1 (4)*, 2001.

B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms (Algorithms and Combinatorics)*. Springer, Berlin, 4 edition, 10 2008.

H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

M. Lai. *Models and algorithms for the empty container repositioning and its integration with routing problems*. PhD thesis, University of Cagliari, 2013.

M. Lai, T. G. Crainic, M. Di Francesco, and P. Zuddas. An heuristic search for the routing of heterogeneous trucks with single and double container loads. *Transportation Research Part E: Logistics and Transportation Review*, 56:108–118, 2013.

A. Langevin, F. Soumis, and J. Desrosiers. Classification of travelling salesman problem formulations. *Operations Research Letters*, 9(2):127–132, 1990.

G. Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, 1992.

S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.

C. Macharis and Y. M. Bontekoning. Opportunities for OR in intermodal freight transport research: A review. *European Journal of Operational Research*, 153:400 – 416, 2004.

F. Meisel and H. Kopfer. Synchronized routing of active and passive means of transport. *OR spectrum*, 36(2):297–322, 2014.

O. Merk, B. Busquet, and R. Aronietis. The impact of Mega-Ships. Case-Specific Policy Analysis. In *OECD International Transport Forum*, 2015.

C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

A. Mingozzi, S. Giorgi, and R. Baldacci. An exact method for the vehicle routing problem with backhauls. *Transportation Science*, 33(3):315–329, 1999.

L. Mingyong and C. Erbao. An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows. *Engineering Applications of Artificial Intelligence*, 23(2):188–195, 2010.

S. Mitrović-Minić and R. Krishnamurti. The multiple TSP with time windows: vehicle bounds based on precedence graphs. *Operations Research Letters*, 34 (1):111–120, 2006.

M. D. Moffitt and M. E. Pollack. Optimal rectangle packing: A meta-csp approach. In *ICAPS*, pages 93–102, 2006.

E. F. Moore. The shortest path through a maze. In *Proceedings of the International Symposium on the Theory of Switching*, volume Part II, page 285âĂŞ292. Harvard University Press, 1959.

R. Namboothiri and A. L. Erera. Planning local container drayage operations given a port access appointment system. *Transportation Research Part E: Logistics and Transportation Review*, 44(2):185–202, 2008.

N. Nordsieck, T. Buer, and J. Schönberger. A three-phase heuristic for a multi-size container transport problem with partially specified requests. Bremen Computational Logistics Group Working Papers, No. 5, University of Bremen, 2016.

N. Nordsieck, T. Buer, and J. Schönberger. Potential of improving truck-based drayage operations of marine terminals through street turns. In M. Freitag, H. Kotzab, and J. Pannek, editors, *Proceedings of LDIC 2016 (5th International Conference on Dynamics in Logistics)*, pages 433–443. Springer International Publishing, 2017.

J. Nossack and E. Pesch. A truck scheduling problem arising in intermodal container transportation. *European Journal of Operational Research*, 230:666–680, 2013.

T. E. Notteboom. Container shipping and ports: an overview. *Review of network economics*, 3(2), 2004.

T. E. Notteboom and J.-P. Rodrigue. Port regionalization: towards a new phase in port development. *Maritime Policy & Management*, 32(3):297–313, 2005.

J. Orlin. A faster strongly polynomial minimum cost flow algorithm. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 377–387, New York, NY, USA, 1988. ACM. ISBN 0-89791-264-0.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. part i: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008a.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008b.

E. Pesch and F. Glover. Tsp ejection chains. *Discrete Applied Mathematics*, 76(1): 165–181, 1997.

S. Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theoretical Computer Science*, 312(1):47–74, 2004.

D. Pisinger and S. Ropke. Large neighborhood search. In *Handbook of metaheuristics*, pages 399–419. Springer, 2010.

D. Popović, M. Vidović, and M. Nikolić. The variable neighborhood search heuristic for the containers drayage problem with time windows. *2012 online conference on soft computing in industrial applications*, pages 1 – 10, 2012.

G. Reinelt. Tsplib-a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991.

N. V. Reinfeld and W. R. Vogel. *Mathematical programming*. Englewood Cliffs, N.J., Prentice-Hall, 1958.

L. B. Reinhardt, S. Spoorendonk, and D. Pisinger. Solving vehicle routing with full container load and time windows. In *Computational Logistics*, pages 120–128. Springer, 2012.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006.

D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, 6 (3):563–581, 1977.

M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995.

J. Schönberger, T. Buer, and H. Kopfer. A Model for the Coordination of 20-foot and 40-foot Container Movements in the Hinterland of a Container Terminal. In D. Pacino, S. Voß, and R.M. Jensen, editors, *ICCL 2013, LNCS 8197*, pages 113–127, Berlin Heidelberg, 2013. Springer-Verlag.

G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139 – 171, 2000.

P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*, 1997.

R. Shostak. Deciding linear inequalities by computing loop residues. *J. ACM*, 28(4):769–779, 1981.

M. Sigurd, D. Pisinger, and M. Sig. *The pickup and delivery problem with time windows and precedences*. Datalogisk Institut, Københavns Universitet, 2000.

K. Smilowitz. Multi-resource routing with flexible tasks: an application in drayage operations. *Iie Transactions*, 38(7):577–590, 2006.

M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

M. M. Solomon and J. Desrosiers. Survey paper-time window constrained routing and scheduling problems. *Transportation science*, 22(1):1–13, 1988.

F. J. Srour, T. Máhr, M. M. De Weerdt, and R. A. Zuidwijk. MIPLIB truckload PDPTW instances derived from a real-world drayage case. Technical report, Erasmus Research Institute of Management (ERIM), 2010.

R. Stahlbock and S. Voß. Operations research at container terminals: a literature update. *Or Spectrum*, 30(1):1–52, 2008.

D. Steenken, S. Voß, and R. Stahlbock. Container terminal operation and operations research-a classification and literature review. *OR spectrum*, 26(1): 3–49, 2004.

S. Sterzik. *Concepts, Mechanisms, and Algorithms to Measure the Potential of Container Sharing in Seaport Hinterland Transportation*. PhD thesis, University of Bremen, 2013.

S. Sterzik and H. Kopfer. A tabu search heuristic for the inland container transportation problem. Chair of Logistics Working Paper No. 3, University of Bremen, 2012.

S. Sterzik and H. Kopfer. A tabu search heuristic for the inland container transportation problem. *Computers & Operations Research*, 40:953–962, 2013.

S. Sterzik, H. Kopfer, and W. Y. Yun. Reducing hinterland transportation costs through container sharing. *Flexible Services and Manufacturing Journal*, 27(2-3):382–402, 2012.

S. Sterzik, H. Kopfer, and J. Funke. Advantages of decoupling containers and vehicles at customers' locations. In J. Dethloff, H.-D. Haasis, H. Kopfer, H. Kotzab, and J. Schönberger, editors, *Logistics Management - Products, Actors, Technology, LM 13 - Proceedings of the German Academic Association for Business Research*, pages 301–312. Springer Verlag, 2015.

É. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673, 1993.

G. D. Taylor and T. S. Meinert. Improving the quality of operations in truckload trucking. *Iie Transactions*, 32(6):551–562, 2000.

G. D. Taylor, G. L. Whicker, and J. S. Usher. Multi-zone dispatching in truckload trucking. *Transportation Research Part E: Logistics and Transportation Review*, 37 (5):375–390, 2001.

P. Toth and D. Vigo. An exact algorithm for the vehicle routing problem with backhauls. *Transportation science*, 31(4):372–385, 1997.

P. Toth and D. Vigo. *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2002. ISBN 9780898715798.

M. R. Van Der Horst and P. W. De Langen. Coordination in hinterland transport chains: a major challenge for the seaport community. *Maritime Economics & Logistics*, 10(1):108–129, 2008.

A. W. Veenstra. Empty container reposition: the port of rotterdam case. In *Managing closed-loop supply chains*, pages 65–76. Springer, 2005.

M. Vidović, G. Radivojević, and B. Raković. Vehicle routing in containers pickup up and delivery processes. *Procedia - Social and Behavioral Sciences*, 20:335–343, 2011.

M. Vidović, M. Nikolić, and D. Popović. Two mathematical formulations for the containers drayage problem with time windows. In *2nd International Conference on Supply Chains ICSC*, 2012.

I. F. A. Vis and R. De Koster. Transshipment of containers at a container terminal: An overview. *European Journal of operational research*, 147(1):1–16, 2003.

M. Wang, B. Liu, J. Quan, and J. Funke. A two-stage iterative solution approach for solving a container transportation problem. In *Logistics Management*, pages 259–271. Springer, 2016.

X. Wang and A. C. Regan. Local truckload pickup and delivery with hard time window constraints. *Transportation Research Part B*, 36:97–112, 2002.

S. Warshall. A theorem on boolean matrices. *Journal of the ACM (JACM)*, 9(1): 11–12, 1962.

H. P. Williams. *Model building in mathematical programming*. John Wiley & Sons, 2013.

World-TSP-Pictures. World tsp pictures. `http://www.math.uwaterloo.ca/tsp/world/pictures.html`, 2004. accessed on 8th October 2015.

Z. Xue, C. Zhang, W.-H. Lin, L. Miao, and P. Yang. A tabu search heuristic for the local container drayage problem under a new operation mode. *Transportation Research Part E: Logistics and Transportation Review*, 62:136–150, 2014.

R. Zhang, W. Y. Yun, and I. Moon. A reactive tabu search algorithm for the multi-depot container truck transportation problem. *Transportation Research Part E: Logistics and Transportation Review*, 45(6):904–914, 2009.

R. Zhang, W. Y. Yun, and H. Kopfer. Heuristic-based truck scheduling for inland container transportation. *OR Spectrum*, 32:787–808, 2010.

R. Zhang, W. Y. Yun, and I. K. Moon. Modeling and optimization of a container drayage problem with resource constraints. *International Journal of Production Economics*, 133(1):351–359, 2011.

R. Zhang, J. C. Lu, and D. Wang. Container drayage problem with flexible orders and its near real-time solution strategies. *Transportation Research Part E: Logistics and Transportation Review*, 61:235 – 251, 2014.

R. Zhang, W.Y. Yun, and H. Kopfer. Multi-size container transportation by truck: modeling and optimization. *Flexible Services and Manufacturing Journal*, 27(2-3):403–430, 2015.

Index

20-foot Truck, 30
40-foot Truck, 30

Adjacent, 15
Admissible Graph, 102
All Pairs Shortest Path Problem, 22
Antichain, 24
Approximate Algorithm, 25
Arc, 15
Arc Capacity, 15
Arc Cost, 15
Assignment Problem, 17, 85
Asymmetric Distance Matrix, 21
Asymmetric Traveling Salesman Problem, 23
asymmetric Traveling Salesman Problem, 32
Autonomous Resource, 30

b-flow, 19
Bipartite Graph, 16, 103
Boltzman Distribution, 26
Branch and Bound, 39
Breadth-First Search, 16

Capacity Constraints, 18
Carrier, 7
Carrier Haulage, 7
Chain, 24
Chassis, 30
Combinatorial Optimization, 13
Combined Truck, 30
Complete Algorithm, 25
Complete Graph, 15
Concorde, 32
Condensation, 16, 103
Connected Component, 15, 83
Connected Graph, 15
Conservative Cost, 21
Constructive Algorithm, 25
Container Leasing Company, 8
Containerization, 2
Cooling Schedule, 26
Cycle, 15

Decision Variable, 14
Depot-direct, 10
Depth-First Search, 16
Deterministic Annealing, 41
Directed Graph, 15
Distance Graph, 95
Distance Matrix, 21
Diversification, 26
Door-to-door Service, 6
Drayage, 1
Drop-and-pick Procedure, 45
Dual Value, 39

End-haulage, 5
Exact Algorithm, 25
Export Container, 31

Feasibility Check, 94, 110
Feasible Solution, 14

Flexible Task, 31
Flow Balance Conservation Constraints, 19
Flow Conservation Constraints, 18
FTPDP structure, 24
Full Truckload, 30, 32, 138
Full Truckload Pickup & Delivery Problem, 24
Full-Twin Assumption, 54, 142

Graph, 15
Greedy Algorithm, 42

Hamiltonian Circuit, 22
Head, 15
Heuristics, 25
Hill Climbing, 26, 81, 130
Hinterland, 1

Import Container, 31
Inbound Container, 31
Incident, 15
Infeasible Problem, 14
Inland Container Transportation Problem, 43
Insertion Heuristic Approach, 93
Integer Programming, 14
Intensification, 26
Interdependence Problem, 33
Intermodal Container Transportation, 1

Jumbo Container, 3

Lagrangian Multipliers, 40
Large Neighborhood Search, 26, 98
Less-than Truckload, 30, 33, 138
Linear Programming, 14
Linear Relaxation, 45
Local Search Algorithm, 25

Main-haulage, 5
Matching, 16
Matheuristics, 27
Maximal Matching, 16
Maximum Flow Problem, 18
Merchant Haulage, 7
Metaheuristics, 26
Minimum Cost Flow Problem, 19, 86
Mixed Integer Programming, 14
Modular Concept Vehicles, 3
Multi-size Inland Container Transportation Problem, 10, 59
Multicommodity Flow Problem, 20, 72
Multiple Traveling Salesman Problem, 23, 76
Multiple Traveling Salesman Problem with Time Windows and Precedences, 34

Negative Reduced Cost, 39
Neighborhood Solution, 25
Neighborhood Structure, 25
Node, 15
Node Balance, 15
Noise, 89
Non-autonomous Resource, 30

Open Routing Problem, 34
Optimum Solution, 14
Outbound Container, 31

Partially Ordered Set, 24
Partition, 14
Path, 15
Perfect Matching, 16
Poset, 24, 102
Pre-haulage, 5
Precedence Constraint, 34, 91

Receiver, 5
Relaxation, 39
Resource, 30

s-t-flow, 18
Sender, 5
Service Time, 35
Set Covering Problem, 25
Set Partitioning Problem, 25, 36, 45
Shipping Company, 7
Shortest Path Problem, 21
Simulated Annealing, 26, 81, 109
Social Constraint, 36
Stay-with Procedure, 39
Street-turn, 10, 38, 60
Strongly Connected Component, 16
Strongly Connected Digraph, 16
Sub-graph, 15
Sub-tour, 23
Symmetric Distance Matrix, 21
Symmetric Traveling Salesman Problem, 23

Tabu List, 42
Tabu Tenure, 42
Tail, 15
Theorem of Dilworth, 24
Theorem of König, 17
Threshold Accepting, 41
Time Horizon, 34
Time Window, 32
Tour, 22
Tractor, 30
Trailer, 30
Transitive Closure, 16, 103
Transitive Digraph, 16
Transport Operator, 7
Transportation Mode, 5
Transportation Problem, 20
Traveling Salesman Problem, 22
Truck, 30
Trucking Company, 7
TSPLIB, 32

Unbounded Problem, 14
Undirected Graph, 15
Unproductive Movement, 1, 9, 29

Vehicle Routing Problem, 23, 45
Vertex Cover, 16

Weighted Graph, 15
Well-defined Task, 31

,